贾　铮　王　韡　雷奇文◎编　著

HTML+CSS网页布局开发指南

- 严格依照CSS最新规范并结合Web标准
- 涵盖**层叠**、**继承**、**盒模型**、**布局**和**定位**等CSS核心技术
- 详细讲解CSS各种样式属性的功能，并配有典型实例
- 代码兼顾多种浏览器，力求获得最好的兼容性
- 介绍**圆角**、**图像替换**、**滑动门**等CSS高级技巧，并提供大量参考资料以便深入学习



清华大学出版社

HTML+CSS 网页布局开发指南

贾 铮 王 韡 雷奇文 编著

清华大学出版社
北 京

内 容 简 介

本书由浅入深、循序渐进地介绍了 HTML、XHTML 和 CSS 的语法、元素、属性的使用方法，介绍了如何使用(X)HTML 和 CSS 编写符合标准的 Web 页面。

本书共分六篇：第一篇从基本概念讲起，包括(X)HTML 基础知识、CSS 基本概念和语法规则；第二篇为 CSS 核心原理，内容涉及选择符、度量、样式层叠与继承，以及如何处理浏览器兼容性问题；第三篇是样式设计，介绍 CSS 如何控制文本、链接、图像、列表、表格和表单元素的样式；第四篇讨论 CSS 布局技术，包括浮动和定位的概念，列举常见的页面布局形式并提供实现方案；第五篇是 CSS 高级主题，介绍 CSS 在 XML/打印媒介中的应用、用户界面元素的样式以及特定于 IE 和 Firefox 浏览器的 CSS；第六篇通过一个较完整的页面实例，介绍如何使用 XHTML 构建页面内容，并综合运用各种 CSS 技术进行页面设计。

本书注重基础、讲究实用、选材精当、深入浅出，适合初、中级读者学习使用，也适合具有 HTML 和 CSS 基本知识的任何网页设计和开发人员阅读及参考。

图书在版编目(CIP)数据

HTML+CSS 网页布局开发指南/贾铮，王韡，雷奇文编著. —北京：清华大学出版社，2008.9
ISBN 978-7-302-18503-1
Ⅰ.H…　Ⅱ. ①贾…　②王…　③雷…　Ⅲ. ①超文本标记语言，HTML—主页制作—程序设计　②主页制作—软件工具，CSS　Ⅳ. TP312　TP393.092

中国版本图书馆 CIP 数据核字(2008)第 136219 号

责任编辑：邹　杰　宋延清
装帧设计：杨玉兰
责任印制：

出版发行：清华大学出版社　　**地　址：**北京清华大学学研大厦 A 座
http://www.tup.com.cn　　**邮　编：**100084
社 总 机：010-62770175　　**邮　购：**010-62786544
投稿与读者服务：010-62776969, c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015, zhiliang@tup.tsinghua.edu.cn
印 刷 者：
装 订 者：
经　　销：全国新华书店
开　　本：190×260　**印　张：**25.25　**字　数：**625 千字
版　　次：2008 年 9 月第 1 版　**印　次：**2008 年 9 月第 1 次印刷
印　　数：1~4000
定　　价：40.00 元

本书如存在文字不清、漏印、缺页、倒页、脱页等印装质量问题，请与清华大学出版社出版部联系调换。联系电话：010-62770177 转 3103　　产品编号：

前　言

本书作者在 9 年前开始接触互联网，体验到网络给人们带来的便捷。又经过 1 年，开始使用 HTML 制作 Web 页面，那时还是表格布局一统天下，使用 CSS 进行布局的人少之又少。作者当然也使用表格布局页面。这样做的结果是，页面充斥着大量的 table 标记和很多控制样式的 HTML 属性和元素，增加了维护工作的难度。后来，随着 Web 标准的逐步推广，要求页面内容与其表现形式分离，而 CSS 就是控制页面表现的语言。于是，作者开始仔细研究 CSS，很快就领略了它的强大功能，后来便一直使用 CSS 进行 Web 开发。

CSS 入门非常容易，它本身并不涉及过于复杂的语法，其代码具有很好的自我说明的特性，即使从未接触过 CSS 的人也能理解大部分代码的含义。

希望通过本书将作者所掌握的 CSS 知识和多年开发中积累的经验与读者分享，帮助读者学习、理解和运用好 CSS。

1. 如何阅读本书

本书采用循序渐进的编写方式，初学者可按顺序阅读每一章的内容。每章的开始部分会告诉你本章所涉及的主要内容，便于快速查阅和了解章节内容。如果读者对某些内容已经掌握，则可以跳过这一章继续阅读。中级读者可以根据需要，挑选某些章节进行阅读。

本书内容严格依照 CSS 规范编写，但并不是对规范的生搬硬套。书中将一些晦涩难懂的专业术语转化为通俗易懂的语言进行讲解，同时配合适当的插图和丰富的代码示例供读者学习和参考。希望读者在阅读过程中能够亲自试验，以加深理解。为了让读者能了解国外相关领域的发展状况，进一步地深入学习，书中还提供了一些英文资料的参考网址。

2. 本书针对的读者

本书主要面向初、中级读者，也适合广大 Web 设计人员和爱好者阅读。对于刚刚步入 Web 设计领域的读者，本书将从最基本的概念开始，逐步介绍使用 CSS 进行 Web 开发的方法。如果读者对 CSS 已有一定的了解，本书也将介绍一些 CSS 开发的高级技巧，并提供进一步学习和研究的参考资料。对于从事 Web 开发工作的读者，本书可作为一本 CSS 参考手册。

3. 致谢

感谢小晖子工作室的翁烨晖、王韡、雷奇文对本书编写所做的贡献，感谢作者家人和朋友的支持，感谢清华大学出版社与编辑，有了他们的帮助本书才能得以顺利出版。

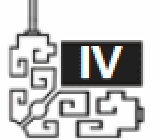
限于作者的水平，书中难免存在错误与疏漏之处，希望读者及 Web 领域的同行给予批评和指正。

编　者

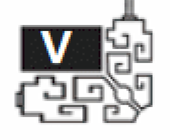
目　　录

第一篇　初识 CSS

第二篇 CSS 核心原理



第三篇 CSS 样式设计



第四篇 CSS 布局技术



第五篇　CSS 高级主题

第六篇　CSS 实战





第一篇　初识 CSS

第 1 章　Web 设计基础

本章首先介绍什么是因特网和万维网，它们之间的区别和联系是什么。接着会介绍一些与 Web 设计相关的基本知识，包括 Web 设计所使用的各种技术、什么是 Web 标准以及它的优势所在。浏览器是用户访问 Web 应用的途径，本章将介绍几款目前较为流行的浏览器，它们各自的发展历程和功能特点。最后让读者了解两种较新的 Web 开发技术。

本章主要内容

- 什么是因特网和万维网，二者是什么关系
- Web 设计技术概述
- 什么是 Web 标准
- Web 标准涵盖哪些内容
- 目前比较流行的浏览器
- Microsoft Silverlight 和 Adobe Flex 技术

1.1　因特网和万维网

1.1.1　何为因特网

各位读者可能对因特网(Internet)这个词不会感到陌生。因特网是由许许多多计算机连接在一起构成的一个计算机网络。在这个网络中，人们可以使用各自的计算机互相传递信息；可以在自己的计算机上存储文件供别人访问；可以利用多台计算机实现分布式应用。Internet 规模庞大，遍及全球。一旦上网，你就与全球无以计数的计算机连成了一体。

1.1.2　何为万维网

有上网经历的读者会注意到，在浏览器的地址栏中经常出现“www”，比如要访问百度网站就需输入“www.baidu.com”。其中的三个“w”是英文 World Wide Web 的缩写，中文译为万维网。万维网是无数个网络站点和网页的集合，它们在一起构成了因特网最主要的部分(因特网还包括电子邮件、Usenet 以及新闻组等应用)。

万维网实际上是多媒体的集合，各个部分由超级链接连接而成。我们通常使用浏览器上网观看的内容，就是万维网的内容。比如在浏览器中输入“www.sina.com.cn”，就可以访问新浪

网站的首页(见图 1-1)。网页也称作 Web 页面或 Web 文档，它包含了文字、图像、动画和一些具有交互功能的控件。



图 1-1　使用浏览器访问新浪网站的首页

1.2　Web 设计技术

Web 设计涵盖的范围相当广，本节只介绍一些构建 Web 文档所需的几项最为基本的技术。

1.2.1　HTML

HTML，英文全称为 HyperText Markup Language，中文译为超文本标记语言。它是用来创建 Web 文档的语言。页面元素是由特定的标记来确定的，这些标记告诉浏览器该如何显示这个页面。所谓超文本，就是一种含有指向其他文档链接的文本，即我们俗称的链接。超文本把存放于不同计算机中的文档链接在一起。访问者不必关心链接所指的内容到底位于何处，只需要在链接上单击一下鼠标，页面就会马上转到所指的文档中去。

1.2.2　XML

XML，即 eXtensible Markup Language(可扩展标记语言)。XML 是一种定义其他语言的语言，是一种元语言(Meta Language)。使用 XML 可以创建自己定制的标记语言，然后使用这种语言对自己的文档进行格式化。

XML 与 HTML 相比，对语法的要求更加严格，也比 HTML 灵活。于是 XML 和 HTML 相



结合产生了 XHTML。

1.2.3 XHTML

XHTML 中的 X 和 XML 中的 X 含义一致，它是可扩展超文本标记语言。XHTML 是由 XML 创建出来的，是对 HTML 的重新改造。XHTML 更健壮、更灵活、更强大，将来可能得到更加完善的支持。

1.2.4 CSS

HTML 可以将内容、结构和格式化信息都包含在同一个 Web 文档中，这样做虽然简单易行，但也存在很多问题。各种信息存放在一起不利于文档的维护，修改起来费时费力。因此 HTML 应只负责文档的内容和结构，而格式化信息交给一套新的语言来完成，这就是 CSS，它也是本书的主角。

CSS 全称为 Cascading Style Sheet，中文译为层叠样式表(也有译作级联样式表的)。其作用就是要控制 HTML 的页面布局和外观样式，使 Web 文档内容结构和表现形式完全分离。

1.2.5 DOM 与 ECMAScript

DOM(Document Object Model，文档对象模型)是 HTML 和 XML 文档的一个应用程序接口(Application Programming Interface，API)。它提供了一种结构化的文档表示方式，从而使开发人员可以修改它的内容以及最终的表达方式。总而言之，它把网页内容与脚本或编程语言联系在一起。

ECMAScript 是由欧洲计算机制造商协会(European Computer Manufacturers Association，ECMA)通过 ECMA-262 予以标准化的脚本程序设计语言。通过 DOM 可对页面中的对象进行访问和操作。目前 Web 页面中经常使用的 JavaScript 和 JScript 与 ECMAScript 兼容，但也包含超出 ECMAScript 的功能。

1.3 Web 标准概述

1.3.1 什么是 Web 标准

Web 标准是由 W3C(World Wide Web Consortium)和其他标准化组织指定的一系列规范的集合，其中包含了上文提到的(X)HTML、XML 和 CSS。Web 标准的目的在于创建一个统一的用于 Web 表现层的技术标准，以便通过不同浏览器或终端设备向最终用户展示信息内容。

1.3.2 使用语义化的(X)HTML

首先我们应该了解(X)HTML 的用途是什么，根据 W3C 的解释，“HTML 是一种在万维网上发布超文本的通用语言……HTML 使用诸如<h1>和</h1>一类的标记来将文本结构化为标题、段落、列表、超文本等。”

所谓语义化，就是指(X)HTML 可以给普通的文本增加结构和含义。比如某段文本是一个普通段落，另一段文本是一个标题或是列表中的一项等。在第 2 章，我们还要详细介绍如何使用语义化标记。

1.3.3 Web 文档的三层结构

1. 结构、表现和行为

Web 文档通常包含三个不同的层次(见图 1-2)。首先是结构层，该层内容是由(X)HTML 编写的文本文档。它包含文档的内容，以及由(X)HTML 添加的语义化的标记。第二层为表现层，这是本书的重点。该层内容是 CSS 样式代码。它描述了文档该以何种样式呈现在访问者面前，样式包括页面各部分的布局、文字段落排版、背景图像和颜色等。第三层是行为层，这里不对它进行过多的介绍。该层定义了文档模型以及如何与用户进行交互，所涉及技术主要是 DOM 以及 ECMAScript 脚本语言。

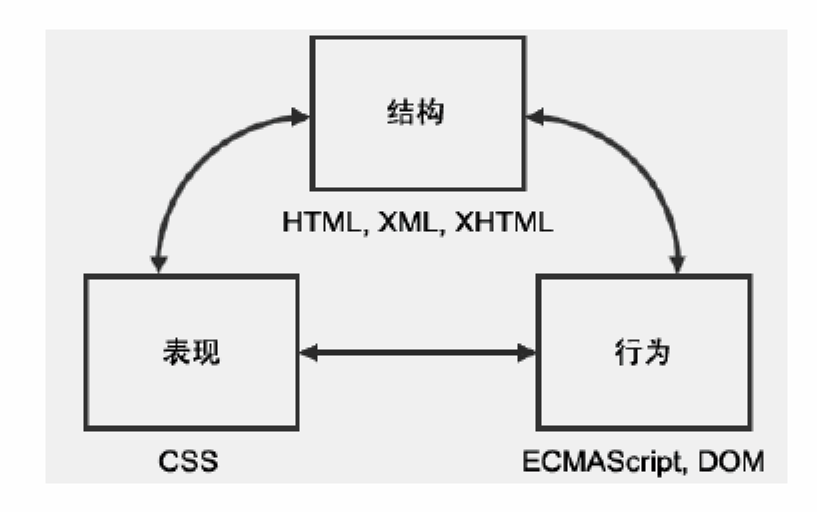


图 1-2 结构、表现和行为及各部分所涉及的技术

2. 模型、视图和控制器

模型、视图和控制器(Model-View-Controller，MVC)模式是面向对象开发中常用的一种设计结构，Web 文档的三层结构也可看作是一种 MVC 结构。模型表示数据结构，也就是(X)HTML 所编写的文档内容。视图表示模型的展现形式，即用 CSS 控制页面的样式。控制器对应着程序的动作行为，因此 Web 中的交互结构可视为控制器。

1.3.4 Web 标准的优势

1. 易于开发和维护

一个大型的网站往往需要很多人员的参与，他们的分工也不同，有负责页面样式设计的，有负责页面编码的，有负责样式编写的等。由于内容结构和表现的分离，不同开发人员可以独



立工作，专注于自己负责的内容。

样式信息和内容是相互独立的，因此同一个样式信息可用于不同的内容中，从而实现代码重用。这种做法可以减少重复编码，加快开发进度。

样式信息重用也使得维护工作大大减轻，只需要修改一小部分样式代码，就可以使所有用到该样式的区域同时改变外观。

2. 高兼容性

由于 W3C 对 Web 标准的推动，越来越多的浏览器支持由 W3C 制定的各种标准，从而使得根据标准编写的 Web 页面在不同的浏览器中均能获得一致的效果。

3. 高灵活性

现在，越来越多的人使用手机或 PDA 访问网站，通常这些设备的屏幕要远远小于 PC 机的显示器。网页的内容和表现的分离使得我们可以针对不同平台和设备应用不同的样式文件，而网页内容无须改动。对于需要打印输出的页面，我们也无须再提供另一份“适合打印”的页面，取而代之的只是新的 CSS 样式。

4. 提高访问速度

内容与表现的分离，使得页面中不再包含冗余的样式代码，而独立出来的样式表可以充分地进行重用，网页整体代码量大大减少。这样不仅能占用更少的网络带宽，提高页面载入速度，同时浏览器也能对页面进行快速解析，显示给用户。

5. 提高用户体验

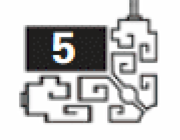
从 Web 应用的角度看，用户体验(User Experience，UE)指的是 Web 应用程序能够提供直观简洁的用户界面、简便的操作以及有效的交互方式。只有当用户亲自使用时才能体验到它们。用户体验包含了多方面的内容，其中一致性、可用性等方面均可通过标准化开发来实现。比如各个页面使用统一的 CSS 样式，利用行为层技术改善交互设计等。

延伸： 傅捷(网络名为阿捷)是国内较早的 Web 标准化执行者，他参与翻译的《网站重构》一书将 Web 标准化思想引入到国内的网页设计行业。在他的网站上有一篇讲述 Web 标准的文章，读者可以访问如下地址来阅读：
http://www.w3cn.org/article/tips/2007/123.html

1.4　浏览器的故事

网页和浏览器是分不开的，用户正是通过浏览器来达到访问网页的目的。浏览器(Browser)是一种软件程序，可以和网络建立连接并与之通信。它可以将存在于万维网中的特定网页获取下来，对网页中的内容进行分析，并按照一定的规则显示出来。

目前主流浏览器有 Internet Explorer、Firefox、Opera、Safari 等，它们适用于各种不同的平台环境。其中最为流行和普及的是 Internet Explorer，它的市场占有率接近于 90%。



1.4.1 Netscape Navigator

1994 年，网景通信公司(Netscape Communications Corporation)推出了 Netscape Navigator(以下简称 NN)浏览器。作为全球第一款商业浏览器，NN 大规模地引领人们开始在网络世界中漫游，正好印证了 Navigator(领航员)这个名字。此后，NN 浏览器迅速走红，一度曾经占据了超过 90%的浏览器市场份额。如图 1-3 所示为以 Netscape Navigator 9 浏览器浏览网页。

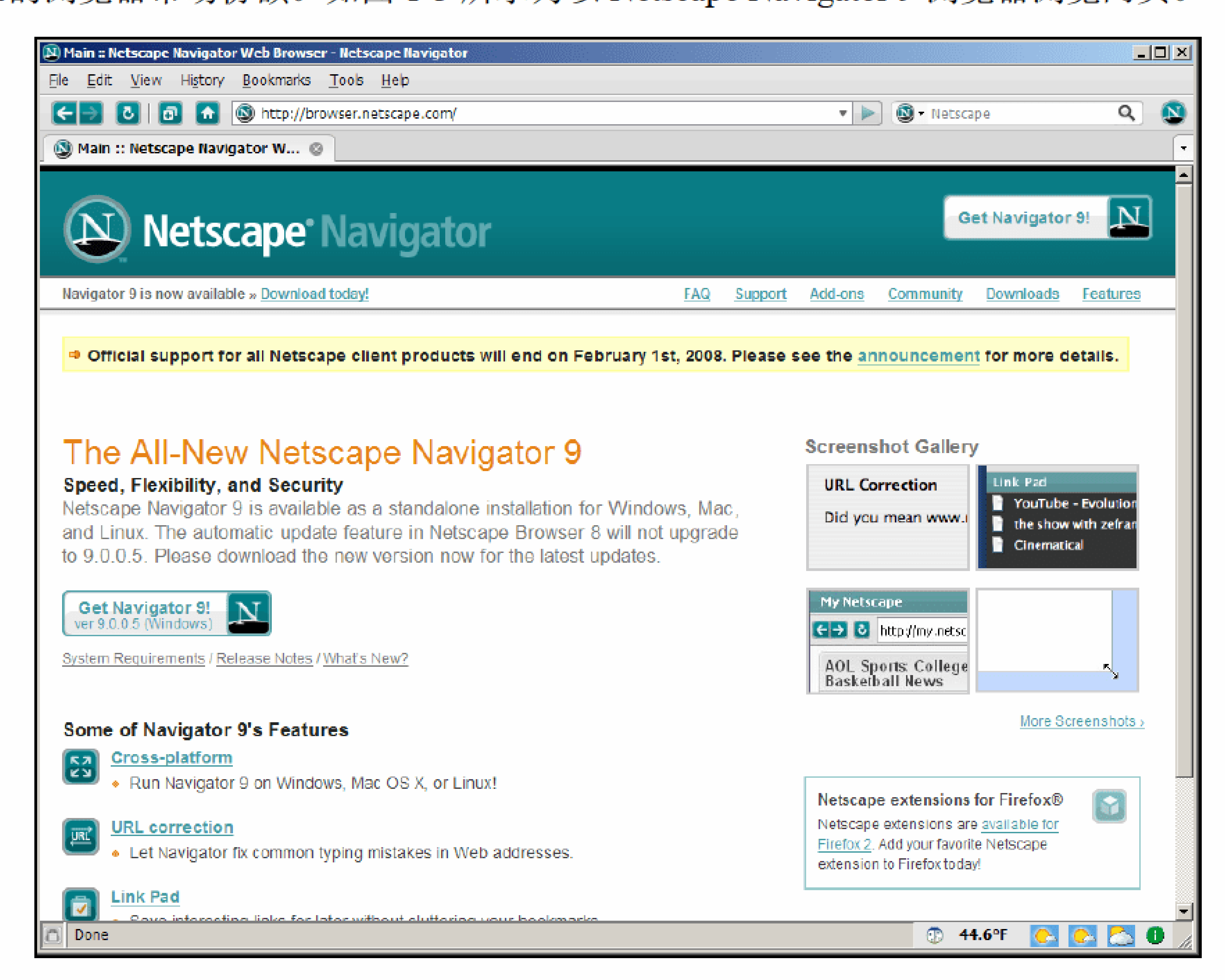


图 1-3　Netscape Navigator 9 浏览器(注意页面中提到停止提供任何技术支持的声明)

NN 浏览器的出现推动了网络的普及，网络的普及也使浏览器市场得到扩大。随着微软公司将自己的 Internet Explorer 浏览器与 Windows 操作系统实行捆绑销售，网景通信公司不得不开始面对挑战。从那时起，NN 浏览器便开始走向没落。

1998 年 11 月网景通信公司被美国在线收购，而美国在线之后又成为时代华纳的一部分。2003 年 7 月 15 日时代华纳解散了网景通信公司。2007 年 12 月 28 日，据网景官方博客消息，NN 浏览器将在 2008 年 2 月 1 日停止更新并且不再提供任何技术支持，1994 年问世的 Netscape Navigator 正式退出历史舞台。

1.4.2 Microsoft Internet Explorer

1995 年，微软公司(Microsoft Corporation)推出了自己的 Internet Explorer(IE)浏览器，并与 Windows 95 操作系统进行捆绑销售，1997 年推出 IE4，增加了活动桌面、频道、FrontPage



Express、Web 发布向导等功能，后来与 Windows 98 捆绑在一起销售。

目前国内使用率最高的是 2001 年发布的 IE6，它集成在 Windows XP 操作系统中。由于 Windows 在操作系统领域占有相当高的市场份额，随之一起销售的 IE 系列浏览器也获得了相当高的市场占有率。但随后微软公司放慢了浏览器开发的步伐，随后的几年中并没有新版本的浏览器问世。鉴于 Firefox、Opera 等浏览器带来新的竞争压力，微软公司后来决定加大对浏览器的开发力度。

2006 年 10 月 18 日，IE7 浏览器发布，它支持标记式浏览、RSS 聚合，增加了快速选项卡、反钓鱼攻击以及页面整体缩放等功能(目前 beta 版的 IE8 浏览器也已经发布)。

专门介绍 IE 浏览器的微软官方页面的地址如下：

```
http://www.microsoft.com/china/windows/products/winfamily/ie/default.mspx
```

该页面当前介绍的 IE 版本为 IE7，如图 1-4 所示。

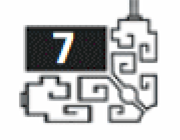


图 1-4　使用 Internet Explorer 7 浏览器

1.4.3　Mozilla Firefox

Mozilla Firefox(http://www.mozilla.org.cn)是由 Mozilla 基金会与数百个志愿者所开发的网页浏览器，如图 1-5 所示。Mozilla 基金会的前身是网景公司内部成立的 Mozilla 组织，为的是在和微软竞争中扭转颓势，Mozilla 组织专门负责维护包括 Netscape Navigator 浏览器在内的多种网络应用软件。后来，Mozilla 组织注册成为非营利机构，并正式更名为 Mozilla 基金会。

Firefox 1.0 正式发布于 2004 年 11 月 9 日，它支持标记浏览、RSS 和多种扩展。2006 年 10 月 24 日，Firefox 2.0 版发布。现在它的 3.0 beta 版也已经发布。



与 IE 浏览器相比，Firefox 更符合 Web 标准，这就是说它会以正确的方式将页面显示出来。由于 Firefox 上的各种插件和工具非常丰富，提高了 Web 开发人员的工作效率。因此它的口碑要好于 IE 浏览器。



图 1-5　Firefox2 浏览器

1.4.4　Opera

Opera Software 开发的 Opera(http://www.opera.com)浏览器(见图 1-6)是一款适用于各种平台、操作系统和嵌入式网络产品的高品质、多平台产品。目前其最新版本为 9.5，该版本浏览器支持同时打开多个选项卡，支持 BT 下载，具有鼠标手势功能和预览页面缩略图功能。



图 1-6　Opera 9 浏览器

Opera 声称自己是“世界上最快”的浏览器，多年以来一直保持着世界第一的称号。它能充分利用缓存机制，快速载入页面。

1.4.5 Safari

Safari(http://www.apple.com/safari)本是苹果电脑中使用的 Mac OS X 操作系统中的默认浏览器，目前也推出了运行于 Windows 平台上的 Safari 3，如图 1-7 所示。



图 1-7 Windows 平台上的 Safari 3 浏览器

1.4.6 其他浏览器

不少使用 QQ 即时通信工具的用户也使用 TT(Tencent Traveler，http://tt.qq.com)浏览器，它是腾讯公司集成在 QQ 中的一款浏览器，目前也提供独立版本的下载。它支持多页面浏览、最近浏览和智能屏蔽等功能。

另一款不得不提的浏览器就是遨游浏览器(Maxthon Browser，http://www.maxthon.cn)，它由北京傲游天下科技有限公司开发，是一款基于 IE 内核的、多功能、个性化、多页面浏览器。它允许在同一窗口内打开任意多个页面，减少了浏览器对系统资源的占用率，同时它又能有效防止恶意插件，阻止各种弹出式广告，加强网上浏览的安全。比较有特色的功能有鼠标手势、超级拖曳、防止页面假死等。

1.5 Web 技术的未来

随着 Web 2.0 应用的不断增多，网页越来越强调丰富的互动体验，逐渐缩小与桌面应用程序之间的使用差距，使其看起来更像是一个应用程序而不是一个普通的网页。对此，Microsoft 和 Adobe 两大公司都提出了相应的解决方案，希望通过出色的用户交互方式和音频、视频等多媒体技术展现下一代的 Web 应用。

1.5.1 Microsoft Silverlight

Microsoft Silverlight(见图 1-8 左)是一个跨浏览器的、跨平台的插件，为网络带来了媒体体验和丰富的交互式应用。Silverlight 提供灵活的编程模型，其 1.0 版本支持 Ajax、JavaScript 和 DHTML 技术，1.1 版本更添加了对 VB.NET、C#、Python、Ruby 等语言的支持，可以很方便地集成到现有的网络应用程序中。

Silverlight 技术提供了一种全新的语言：XAML(XML Application Markup Language)，并提供相应的编程接口，供 Web 设计人员和程序开发人员共同使用。

Silverlight 旨在创建一种与桌面程序相似的 Web 应用，最终使二者达到一致。

1.5.2 Adobe Flex

Adobe Flex(见图 1-8 右)是一个高效、开源的框架，用来构建和维护表现力丰富的 Web 应用程序，它支持各种主流浏览器和不同的平台。Flex 提供一种现代的、标准化的语言和编程模型，支持常见的设计模式。其中，MXML 是一种基于 XML 的语言，用来创建用户界面和行为。ActionScript 3 是一种面向对象的编程语言，通过它来实现客户端的逻辑。



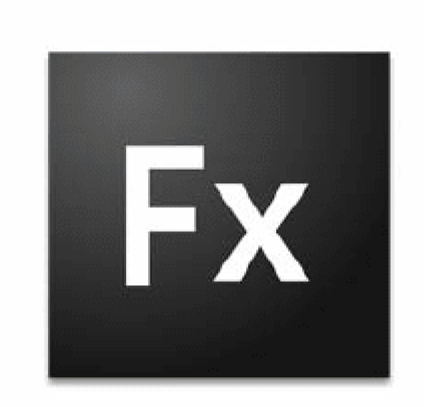
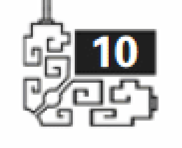


图 1-8 Microsoft Silverlight 与 Adobe Flex

使用 Flex 开发的应用程序可运行于安装了 Adobe Flash Player 的浏览器中，或通过 Adobe AIR 运行于桌面。这使得 Flex 应用程序可以跨浏览器、跨平台执行。

Flex 可以使 Web 开发人员快速地创建 Web 应用，它提供了高效的开发环境(Adobe Flex Builder、Adobe Live Cycle Data Services ES)和一些高级数据服务的集合。



延伸： RIA(Rich Internet Application，富因特网应用程序)指的是一种具有丰富交互能力的应用程序，这是相对于传统的网页而言的。RIA 能提供比基于(X)HTML 的网络应用更加健壮、更加灵活和更加令人感兴趣的可视化特性。

1.6 小 结

因特网的出现使得人们可以方便地访问其他计算机中的资源，而万维网则使资源能够以网页的形式供他人浏览。

HTML 和 XHTML 是一种用来创建 Web 文档的语言，文档中可以包含指向其他资源的链接，通过链接可以访问其他资源。CSS 用来控制页面的外观，DOM 和脚本语言可实现页面的交互。

Web 标准由一系列的规范组成，使用 Web 标准进行网页开发能在诸多方面体现出优势。

浏览器与网页密不可分，本章介绍了一些主要浏览器的发展及功能特点。

本章最后介绍了两种较新的 Web 开发技术，Microsoft Silverlight 和 Adobe Flex。

下一章将介绍 HTML 和 XHTML 的基础知识，为进一步学习 CSS 打好基础。

第 2 章　HTML 和 XHTML 基础

本章将向读者介绍 HTML 和 XHTML 的基本知识，将从编写一个简单的 Web 页面开始，逐步讲解标记、元素、属性等有关概念，接着介绍(X)HTML 文档的基本结构及文档树的概念，最后说明如何编写符合标准的(X)HTML 文档。

对(X)HTML 已经很熟悉的读者可以跳过本章，继续阅读后面的内容。

本章主要内容

- (X)HTML 中的标记、元素和属性的概念
- (X)HTML 中的注释、空白和特殊字符
- (X)HTML 的文档基本结构组成
- 文档树的概念
- 如何编写标准的(X)HTML 代码

2.1　页面设计初探

2.1.1　编写第一个 Web 页面

在大多数介绍编程语言的书籍中，第一个程序的基本功能几乎都是输出那耳熟能详的一句话：Hello, World!。我们的程序也与此类似，本小节将带领读者完成第一个 Web 页面的制作，功能非常简单，在页面输出以下几个字符：

```
Hello, XHTML and CSS!
```

首先打开 Windows 的记事本(或者任何你喜欢使用的文本编辑器)，输入以下内容：

```
<html>
<head>
<title>第一个 Web 页面</title>
</head>

<body>
<h1>Hello, XHTML and CSS!</h1>
</body>
</html>
```

把此文件保存为 first.html，注意文件的后缀名为“html”而非“txt”(见图 2-1)。

然后在浏览器中打开这个文件，产生如图 2-2 所示的效果。

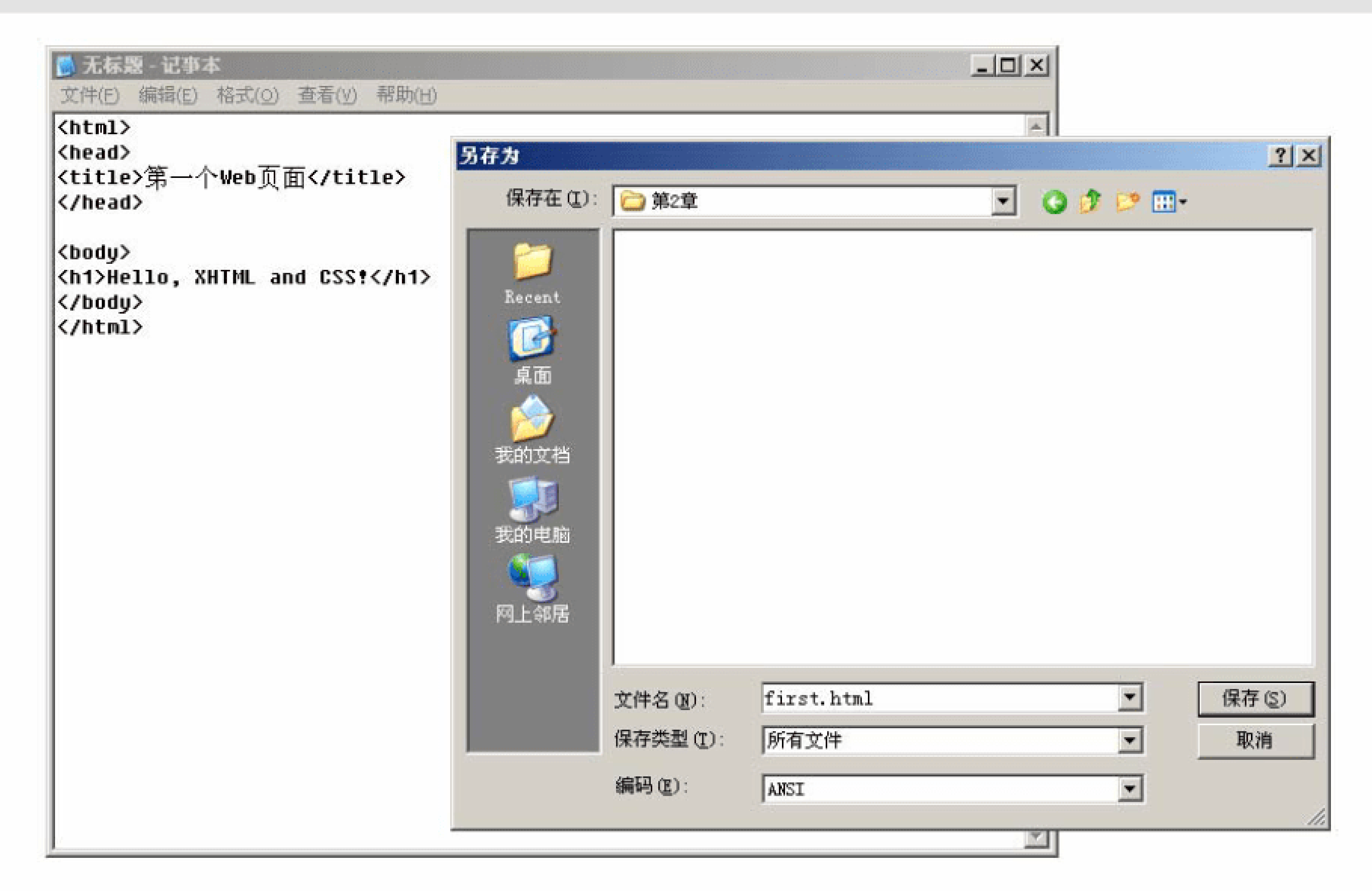


图 2-1　保存文件

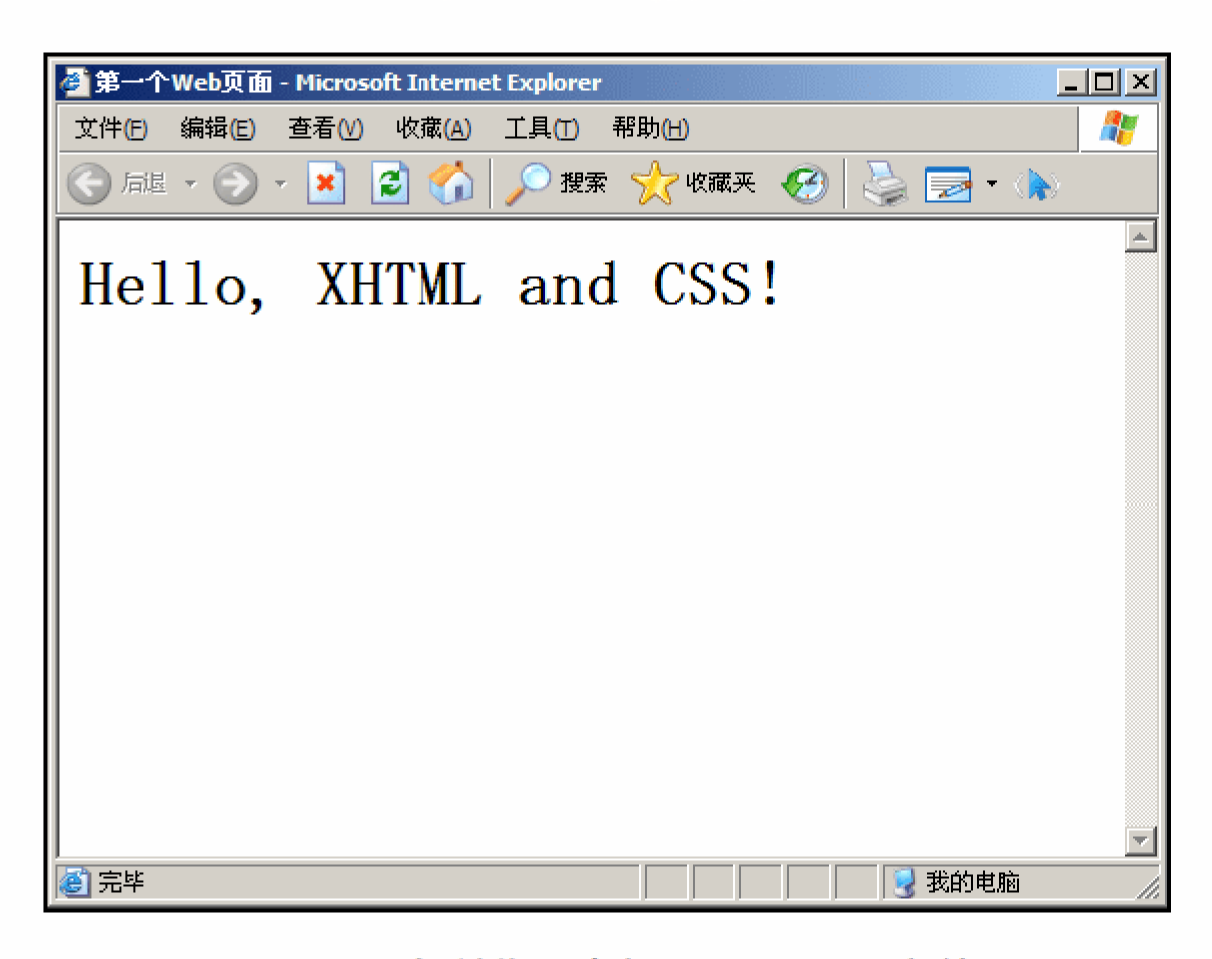


图 2-2　在浏览器中打开 first.html 文件

2.1.2　first.html 中都有些什么

在我们编写的 HTML 文件中，开始和结束的位置包含这样的内容：<html>和</html>。这是一对标记(Tag)，用来告诉浏览器文件中包含的内容是以 HTML 文档方式编写的。

在 HTML 和 XHTML 文档中主要包含两个部分：头(head)部和主体(body)，在我们的范例中同样包含了这两个部分。头部用如下标记表示：<head>和</head>。头部一般放置网页标题、与该网页有关的元信息、脚本代码和样式代码等。本例中我们只放置了一个网页标题(title)，它由标记<title>和</title>表示，位于这两个标记之间的文字会显示在浏览器的标题栏中(见图 2-2)。主体部分由标记<body>和</body>表示，主体区域包含 HTML 文档的内容，这些内容会



显示在浏览器中。在我们的范例中，主体区域只包含一对<h1>和</h1>标记及标记之间的文字。<h1>和</h1>标记表示网页内容中的标题，h 是英文 heading(标题)的首字母，后面的 1 表示它是第一级标题，标记之间的文字就是该标题的内容，并能显示在浏览器窗口中(见图 2-2)。

2.2 标记、元素和属性

2.2.1 标记、内容和元素

1. 标记

(X)HTML 的关键内容之一就是标记(Tag)，任何标记语言也是如此。标记是一些符号，用来区分文档中的不同部分，同时还能标识出内容的类型。浏览器通过识别这些标记来正确地处理不同的内容。标记的描述性名称一般都能体现出其所包含的内容类型，比如标题、段落、列表、图像等。

(X)HTML 中的标记是由一对尖括号“<”和“>”括起来的，用来区分其他普通的文本。左尖括号“<”表明一个标记的开始，紧跟着是标记名称(Tag Name)，或者称作元素名称(Element Name)，最后以右尖括号“>”作为标记的结束。比如下面这个标记就表示一个段落(Paragraph)的开始：

```
<p>
```

在 HTML 规范中，标记名称是不区分大小写的，因此<body>、<BODY>和<bOdY>表示相同的标记，但是 XHTML 规范中要求标记名称要采用小写的形式。

大部分标记是成对出现的，由起始标记来表示一段内容的开始，由结束标记表示结束。结束标记比起始标记多一个斜杠“/”，它位于标记名称之前。起始标记和结束标记之间包含的部分称为内容(Content)。下面这段代码表示了一个完整的段落：

```
<p>这是一段文字。</p>
```

2. 元素

起始标记、结束标记和其中的内容构成了一个完整的元素(Element)，元素可视为一个容器(Container)，将内容包含在其中。元素是(X)HTML 文档基本的组成单位。一些空元素(Empty Element)本身不包含也不可能包含任何内容，这些元素可以没有结束标记。但是在 XHTML 中，这样的元素需要是自我结束(Self-closed)的。在起始标记右尖括号之前加一个斜杠“/”即可。比如表示换行标记本身不包含其他内容，它的标记写法如下：

```
<br />
```

img 元素用来向页面中添加图像文件，它也不包含任何内容，因而也是自我结束的：

```
<img src="logo.gif" alt="logo" />
```

像 p、h1、h2 这样能够包含内容的元素被称为非替换元素(Non-replaced Elements)，而像 img 这样的元素被称为是替换元素(Replaced Elements)。替换元素的含义是，当文档显示在浏览器



中时，替换元素的位置上将会出现页面之外的资源(如图像、Flash 动画等)。

(X)HTML 元素允许嵌套(Nested)出现，比如以下代码中 p 元素除了包含文字外，还包含一个 em 元素(Emphasize，表示强调)：

```
<p>这个内容<em>非常重要</em>。</p>
```

嵌套必须正确，元素的起始标记和结束标记之间不能只包含一个起始标记或者一个结束标记，比如下面的写法就是错误的：

```
<p>这个内容<em>非常重要。</p></em>
```

这里，p 元素中只包含了一个 em 元素的起始标记，em 元素中也只包含了一个 p 元素的结束标记。

2.2.2　元素类型

(X)HTML 元素可以划分为两大类：块级(Block-level)元素和内联(Inline)元素。块级元素所包含的内容独占一行，比如段落元素 p、列表元素 li、标题元素 h1~h6 等。请看如下代码：

```
<html>
<head>
<title>块级元素</title>
</head>

<body>
<h2>块级元素</h2>
<p>块级元素会独占一行。</p>
<p>块级元素前后都产生换行。</p>
</body>
</html>
```

打开 Windows 记事本，输入以上代码并保存为 block.html。然后在浏览器中打开它，会出现如图 2-3 所示的效果。我们看到 h2 元素和两个 p 元素中的文字都单独占用一行的空间。

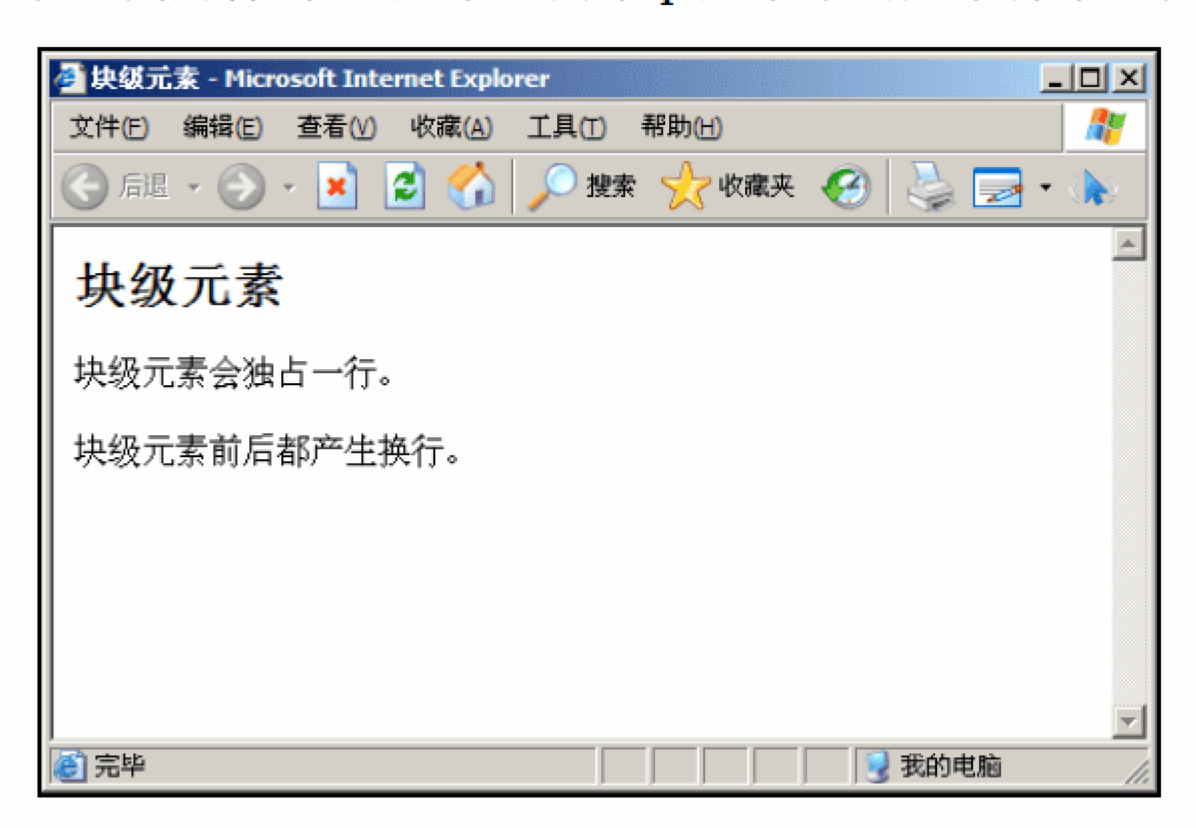


图 2-3　块级元素的显示

内联元素会紧挨着相邻元素，前后不产生换行。比如表示强调内容的 em 元素、表示链接的 a 元素。请看如下示例：

```
<html>
<head>
<title>内联元素</title>
</head>

<body>
<span><em>内联元素</em>会紧挨着相邻的元素，</span>
<span>元素前后不产生换行。</span>
</body>
</html>
```

在记事本中输入以上代码并保存为 inline.html，打开后会产生如图 2-4 所示的效果。

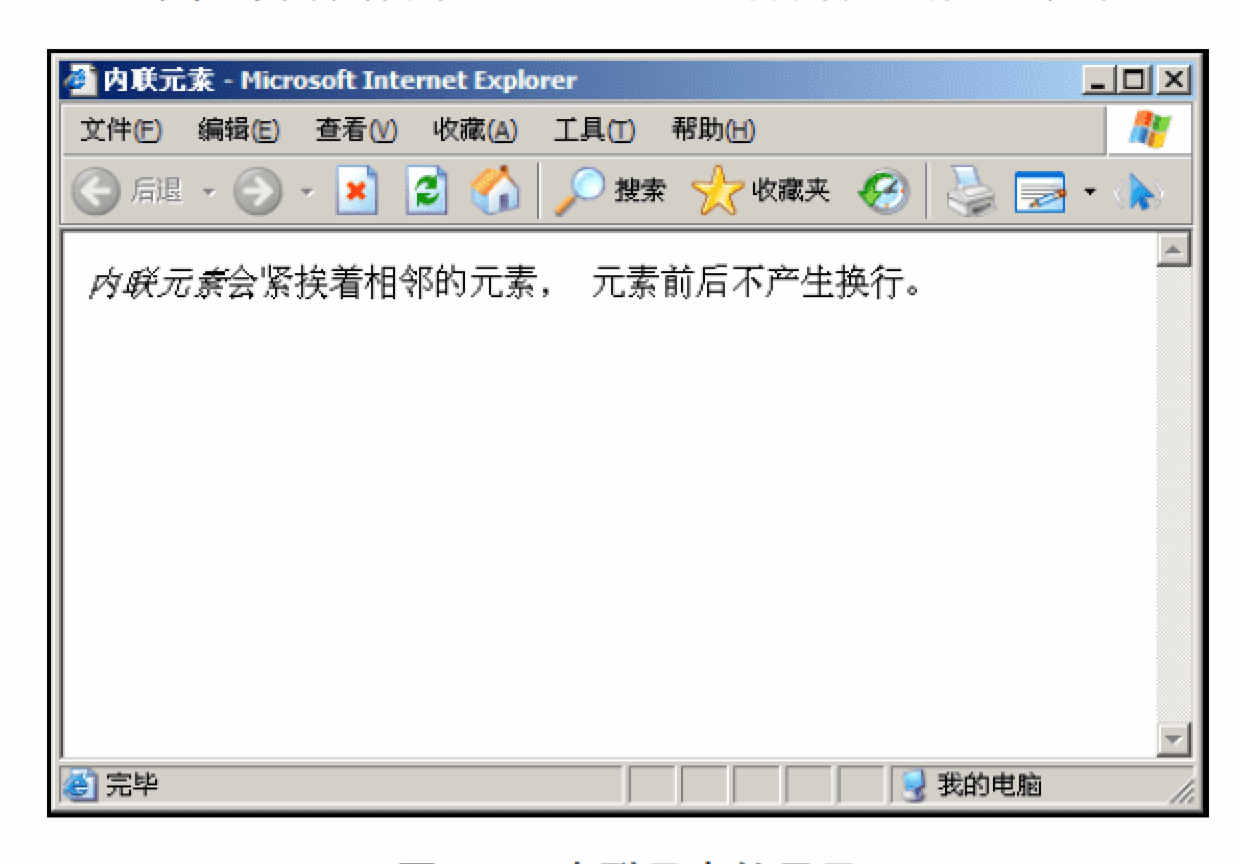


图 2-4 内联元素的显示

(X)HTML 的元素都是有含义的，元素的类型和元素的含义是一致的，比如一个段落通常和其他段落之间有换行，因此 p 元素属于块级元素是非常合理的。

块级元素允许包含其他的块级元素、文本和内联元素(p 元素例外)。而内联元素只允许包含文本和其他内联元素。

2.2.3 属性

每个元素的起始标记内可以添加属性(Attribute)，用来提供一些元素的附加信息。属性由属性名加属性值构成，多个属性之间用空白相隔。比如：

```
<p class="normal" id="p1">这是一段文字。</p>
```

p 元素包含了两个属性，其中 class 属性的值是 normal，id 属性的值是 p1。通过添加属性，就可以达到把该段落同其他段落区分开来的目的。XHTML 要求属性名和预定义的属性值必须小写，且属性值必须由引号(单引号或双引号)引起来。

图 2-5 总结了目前我们学习过的一些基本概念。



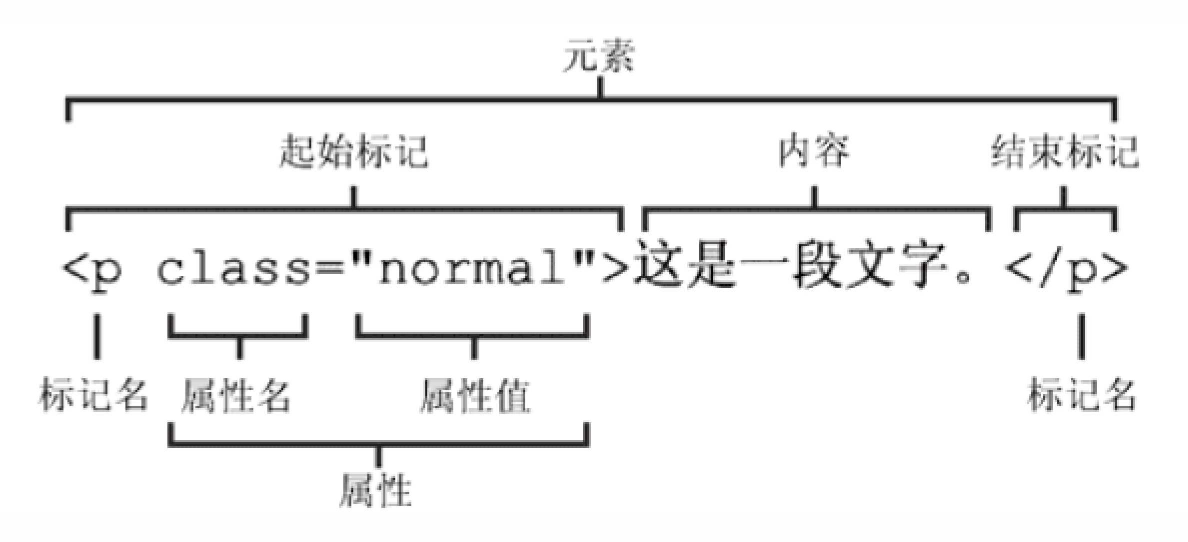


图 2-5　标记、元素和属性

不同的(X)HTML 元素可以拥有不同的属性，但是以下 4 个属性是每个元素都可以使用的，它们是(X)HTML 的核心属性。

1. id 属性

id 属性用来给元素分配一个唯一的标识符。注意标识符必须是唯一的，多个元素不能共用同一个 id 属性值。例如：

```
<p id="myParagraph">本段文字被指定了 id 属性</p>
```

上面的 p 元素的 id 属性值为 myParagraph，后面会讲到，我们通常利用这个 id 属性值给某个特定的元素指定 CSS 样式。比如：

```
p#myParagraph{
    color:blue;
}
```

这是一条 CSS 的样式规则，它将 id 为 myParagraph 的 p 元素中的文字颜色设为蓝色。由于 id 属性需要满足唯一性的要求，因此该规则只对这个 p 元素有效。

2. class 属性

class 属性指明元素属于的一个或多个类别。与 id 属性不同，多个元素可以拥有相同的 class 属性。而且一个元素也可以同时属于多个类，多个类名之间用空格隔开。比如：

```
<p id="myParagraph" class="highlighted">本段文字被指定了 class 属性</p>
```

p 元素除了被指定了一个唯一的 id 值外，还属于 highlighted 类。由于多个元素可以属于同一个 class，CSS 通常利用这个属性给某些特定的元素添加样式。比如：

```
.highlighted{
    font-weight:bold;
}
```

上面的 CSS 规则将属于 highlighted 类的所有元素的文字以粗体显示。

3. style 属性

style 属性指定元素的 CSS 样式。以这种方式添加的样式称作内联样式(Inline Style)，比如：

```
<p style="color:red; font-size:12px;">style 属性用来给元素添加 CSS 样式。</p>
```

以上代码中的 style 属性将 p 元素内的文字设为红色，字体大小为 12px。尽管使用 style 属

性添加样式是可以的，但是实际编写网页样式时不使用这种方式，因为它将样式信息和文档内容混合在一起，使代码变得混乱、难以管理。

4. title 属性

title 属性给元素提供一个标题。大多数浏览器会以提示框(Tooltip)的形式显示 title 属性值的内容。比如：

```
<p title="这是一个 p 元素">这是一段文字。</p>
```

如图 2-6 所示为 IE 浏览器中的效果。

图 2-6　title 属性的作用

2.3　注释、空白和特殊字符

2.3.1　注释

(X)HTML 文档中可以包含注释信息，这些内容不会被浏览器显示在页面上，只有在查看页面源文件时才会浏览到注释信息。(X)HTML 文档的注释以符号“<!--”开始，以符号“-->”结束。例如：

```
<!--主体区域开始-->
<body>
...
```

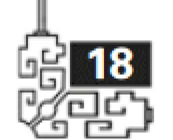
在以上代码中，body 元素之前的内容就是注释信息，它不会被浏览器显示出来。当页面内容变得庞大时，适当添加注释信息有助于编程人员理解代码，提醒协同的开发人员代码的修改和更新。注意注释不能嵌套出现。

2.3.2　空白的处理

为了使(X)HTML 文档便于阅读，我们通常会在编写代码的过程中采用某种样式风格。比如适当添加换行符、制表符(Tab 键)等。比如我们通常会把代码写成如下样式：

```
<html>
<head>
<title>文档标题</title>
</head>

<body>
```



```
<div>
    <p>文档内容</p>
</div>
</body>
</html>
```

而不是下面这样：

```
<html><head><title>文档标题</title></head><body><div><p>文档内容</p></div>
</body></html>
```

以上两种写法最终产生的页面效果是一致的，但是第一种更利于阅读，能更加清晰地展示文档内容和结构。之所以这两种写法不同的代码具有一致的效果，是因为浏览器会忽略文档中多余的换行符、制表符和空格符，将它们都视为一个空格符。换行符、制表符和空格符统称为空白(White Space)。本例中，对于 body 元素之外的元素，它们的内容并不显示在浏览器中，因此不存在显示的问题，对于 body 元素之内的元素，空白都是在元素之间产生的，且元素类型都是块级元素，空白不影响它们的显示效果。但是如果空白添加在元素内容之中，或者添加在内联元素之间，则浏览器会在相应的位置显示一个空格符，请看如下示例：

```
<html>
<head>
<title>空白处理 1</title>
</head>

<body>
<div>
    <p><span>换行符、制表符和空格符</span><span>统称为空白。</span></p>
</div>
</body>
</html>
```

p 元素中文字之间、span 元素之间都没有空白，这时浏览器的显示效果如图 2-7 所示。

换行符、制表符和空格符统称为空白。

图 2-7　元素内容中和内联元素之间没有空白

现在将代码做如下调整，添加一些空白：

```
<html>
<head>
<title>空白处理 2</title>
</head>

<body>
<div>
    <p><span>换行      符、制表符


        和空格符</span>
```

```
<span>统称为                    空白。</span></p>
</div>
</body>
</html>
```

可以看到，我们在 p 元素内容之间和 span 元素之间添加了一些空白(一个回车符不会产生空格，所以我们在 span 元素中添加了两个连续的回车符)，浏览器的显示效果如图 2-8 所示。

换行 符、制表符 和空格符 统称为 空白。

图 2-8　浏览器会将多个空白显示为一个空格符

由此我们得知，空白的添加既要保证代码清晰同时又不破坏页面原有的显示效果。如果我们希望页面的样式和代码中的效果一致呢？可以使用(X)HTML 的 pre 元素，该元素中的内容都将严格按照代码中出现的字符效果显示。比如我们将上例 body 元素之间的内容改写为：

```
<pre>
    <p><span>换行    符、制表符



        和空格符</span>

<span>统称为                    空白。</span></p>
</pre>
```

则浏览器的显示效果如图 2-9 所示(图 2-9 中为使用 IE 浏览器的效果，注意使用 pre 元素后，其中文字大小也发生了变化，而在 Firefox 浏览器中，字体的大小不会发生变化)。

```
换行      符、制表符


                  和空格符

统称为                      空白。
```

图 2-9　pre 元素中的内容将严格按照代码中的字符显示(IE 浏览器)

2.3.3　特殊字符

有时候我们需要在页面中放置一些特殊的字符，比如表示版权的符号“©”、表示注册商标的符号“®”等。

此外，浏览器有时会把小于号“<”和大于号“>”当作是一个标记的开始和结束标志，而不会显示在页面上。这可能会导致页面的某些内容显示不正常。

那么这些符号该如何显示呢？

(X)HTML 提供了若干字符实体(Character Entities)来表示这些特殊字符，字符实体由“&name”或者“&code”来表示，表 2-1 列出了一些常用的字符实体。



表 2-1　常用的字符实体

&name	&code	符　号	说　明
&	&	&	表示和符号
<	<	<	小于号
>	>	>	大于号
™	™	™	商标
			不产生换行的空格
©	©	©	版权符号
®	®	®	注册商标

延伸： 以下网址提供了 HTML4 和 XHTML1 所使用的字符实体的参考，有兴趣的读者可以访问这个链接获得更多关于字符实体的知识。网址：
http://www.cookwood.com/html/extras/entities.html

2.4　(X)HTML 文档结构

在本章 2.1 节中，我们通过编写一个简单的 Web 页面初步了解了(X)HTML 文档的基本组成部分，在本节中将详细讲解 HTML 和 XHTML 的文档结构，以及它们之间的不同之处。

2.4.1　文档类型声明

通俗地讲，文档类型声明(Document Type Declaration，DTD)的作用就是告知浏览器文档中包含的内容属于什么类型，以便用相应的规则来解释和处理各种标记。若使用了不正确的文档声明或根本不进行文档声明，浏览器会按自己的方式进行解析，可能会产生预想不到的页面效果。因此一个标准的(X)HTML 文档必须包含相应的文档类型声明，它位于文档的最开始处。

延伸： 文档类型声明错误或缺失的(X)HTML 文档会使浏览器工作于兼容模式(Compatibility Mode)或怪异模式(Quirks Mode)。此时的浏览器会模拟旧版本的行为对页面进行分析和处理。著名的 CSS 专家 Eric Meyer 在下面这个网址中展示了不同浏览器是如何根据文档类型声明来选择相应工作模式的：
http://meyerweb.com/eric/dom/dtype/dtype-grid.html

(1) HTML 4.01 定义了三种文档类型：严格型(Strict)、过渡型(Transitional)和框架型(Frameset)。

① 严格型要求不能使用任何表现层的属性和元素，页面样式全部交给 CSS 控制。严格型的 DTD 代码如下：

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
```

② 过渡型要求比较宽松，允许使用表现层的属性和元素，当用户浏览器不支持 CSS 样式时，可以使用这种方式控制页面样式。过渡型的 DTD 代码如下：

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
```

③ 框架型用于那些使用框架的页面，除了允许框架替代 body 外，框架型和过渡型的要求是一致的。框架型的 DTD 代码如下：

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
"http://www.w3.org/TR/html4/frameset.dtd">
```

(2) XHTML 1.0 也提供了三种文档类型声明可供选择：严格型、过渡型和框架型。

① 严格型要求不能使用任何表现层的属性和元素，例如表示换行的 br 元素、表示背景色的 bgcolor 属性等都不允许使用。严格型的 DTD 代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

② 过渡型的要求非常宽松，它允许继续使用 HTML 4.01 的标识，但是要符合 XHTML 的语法。过渡型的 DTD 代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

③ 框架型专门针对框架页面设计使用，如果你的页面中包含有框架，则需要采用这种 DTD。框架型的 DTD 代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
```

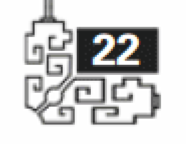
严格地讲，XHTML 只能包含定义文档内容的结构，任何属于表现层的属性和元素都不允许出现，因此应该使用严格型的文档类型声明。但是严格型的声明会大大增加代码编写工作的难度，所以一般情况下使用过渡型的声明。本书中的所有示例均采用过渡型的 XHTML 文档编写。

延伸： 文档类型声明描述了文档中允许出现的语法和词汇表，也就是定义了文档的整体结构以及文档的语法。简而言之，它规定了一个语法分析器在解释一个有效的 XML 文档时需要知道的所有规则。读者若想了解更多关于文档类型声明的知识，请参考一些 XML 方面的书籍。

2.4.2 文档元素

html 元素表示了一个(X)HTML 文档，标记<html>和</html>之间的内容都属于这个(X)HTML 文档。html 元素包含文档头和文档体两部分内容。

在使用 Dreamweaver 创建 XHTML 文档时，可以看到 html 元素会包含这样一个属性：



```
xmlns="http://www.w3.org/1999/xhtml"
```

这个属性定义了该 XHTML 文档所属的命名空间，简单地讲，命名空间的作用就是通过指定一个唯一的属性值(这里使用的是一个 URL)，把本类型的文档同其他类型的文档区分开来，防止冲突的发生。本书中的示例代码都会包含这个属性。

2.4.3　文档头

文档头包含一些关于本(X)HTML 文档的属性，比如文档标题、与其他文档之间的关系等。位于此区域的大部分元素对于页面的访问者来说都是不可见的。

文档头由 head 元素表示，起始标记<head>紧挨着<html>标记之后，<head>和</head>之间定义了文档头部区域。head 元素中通常会包含其他一些元素，比如 title、meta 等。这里向读者介绍一些常见的元素。

1. title 元素

title 元素用来定义文档的标题。浏览器一般会在标题栏位置显示这个标题的内容，下面的文档中包含了一个 title 元素：

```
<html>
<head>
<title>CSS 网页设计开发</title>
</head>
</html>
```

如图 2-10 所示为不同浏览器对标题栏的处理。



图 2-10　title 元素在 IE6(左)、Firefox(中)和 Opera(右)浏览器中的显示情况

2. meta 元素

meta 元素用来给文档添加附加的数据，与 title 元素不同的是，meta 元素不包含任何内容，通过元素属性来达到添加附加数据的目的。meta 元素中的属性包括：charset、content、dir、http_equiv、lang、name 和 scheme。

charset 属性定义了文档所使用的字符集，简体中文的字符集为 gb2312，一般英文页面的字符集为 utf-8。但是这种方式是 IE5 以及更早版本的浏览器支持的方式，这里建议使用 content 属性来给文档添加字符集。

name、http-equiv 和 content 属性给文档添加了一系列的名/值对，name 属性和 http-equiv 属性都可用来定义名字，content 属性定义值部分。一些网站会添加如下的 meta 元素：

```
<meta name="keyword" content="购物, 时装, 饮食" />
```

从 name 属性的名称就能看出，该 meta 元素给文档增加了一些关键字的信息。

使用 Dreamweaver 创建的 XHTML 文档会包含以下 meta 元素：

```
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
```

这个 meta 元素告诉浏览器，页面内容类型是 text/html，使用的字符集是 gb2312，即简体中文。

3. script 元素

script 元素可以给文档添加脚本代码。我们既可以将脚本代码写在 script 元素内容中，又可以通过 script 元素的 src 属性指定一个外部的脚本文件。比如：

```
<script language="javascript" type="text/javascript" src="js/load.js">
</script>
```

language 属性和 type 属性分别指明脚本语言是 JavaScript，类型是 text/javascript。

4. style 元素

style 元素用来给文档添加 CSS 样式。样式代码位于<style>与</style>之间，后面会讲到，这种方式添加的样式称为嵌入样式(Embedded Style)。比如：

```
<style type="text/css">
p{color:red;}
</style>
```

其中，type 属性表明它的类型是 text/css。

5. link 元素

link 元素用来定义本文档和其他一些文档之间的关系。该元素也可以用来添加 CSS 样式，这种方式添加的样式称为外部样式(External Style)。比如：

```
<link href="styles/global.css" rel="stylesheet" type="text/css">
```

其中，href 属性定义了 link 元素的目标位置，这里是一个 CSS 样式文件。rel 属性定义了本文档与目标位置之间的关系，属性值 stylesheet 表明目标位置的文档是本文档的样式表单。type 属性描述了目标位置的文档类型，这里 text/css 表示它是一个 CSS 文档。

2.4.4 文档体

文档体是真正放置页面内容的区域。<body>和</body>标记之间的部分就是文档的主体部分。最简单的文档可能只包含一些文字，复杂的文档可能包含许多元素，比如区域、表格、段落、标题、图像、动画等。下面我们将介绍一些常见的(X)HTML 元素，部分元素将会放在后续章节中详细讲解。

1. 段落

可能读者在上小学的时候就听老师讲过，段落是根据文章或事情的内容、阶段划分的相对独立的部分。类似地，(X)HTML 文档也有段落(Paragraph)元素，它使用 p 元素表示一个段落。



在显示上，每个 p 元素之间会存在一定的距离。请看以下代码：

```
<p>HTML 是 Hypertext Markup Language 的缩写，即超文本标记语言。它是用于创建可从一个平台移植到另一平台的超文本文档的一种简单标记语言，经常用来创建 Web 页面。HTML 文件是带有格式标识符和超文本链接的内嵌代码的 ASCII 文本文件。
</p>
<p>HTML 是制作网页的基础，我们在网络营销中讲的静态网页，就是以 HTML 为基础制作的网页，早期的网页都是直接用 HTML 代码编写的，不过现在有很多智能化的网页制作软件(比如 Dreamweaver)通常不需要人工去写代码，而是由这些软件自动生成的。尽管不需要自己写代码，但了解 HTML 代码仍然非常重要，是学习网络营销与电子商务的技术基础知识。
</p>
```

以上代码出现了两个 p 元素，如图 2-11 所示为浏览器的显示效果。

HTML是Hypertext Markup Language的缩写，即超文本标记语言。它是用于创建可从一个平台移植到另一平台的超文本文档的一种简单标记语言，经常用来创建Web页面。HTML文件是带有格式标识符和超文本链接的内嵌代码的ASCII文本文件。

HTML是制作网页的基础，我们在网络营销中讲的静态网页，就是以HTML为基础制作的网页，早期的网页都是直接用HTML代码编写的，不过现在有很多智能化的网页制作软件（比如 Dreamweaver）通常不需要人工去写代码，而是由这些软件自动生成的。尽管不需要自己写代码，但了解HTML代码仍然非常重要，是学习网络营销与电子商务的技术基础知识。

图 2-11　p 元素的显示效果

p 元素属于块级元素，每个 p 元素的前后都会产生一个换行。

2. 分级标题

(X)HTML 按照标题的不同等级，提供了 6 个表示标题(Heading)的元素：h1、h2、h3、h4、h5 和 h6。在显示上，不同级别的标题会以不同的样式显示。比如：

```
<h1>一级标题</h1>
<h2>二级标题</h2>
<h4>四级标题</h4>
<h6>六级标题</h6>
```

在浏览器中的显示效果如图 2-12 所示。h1~h6 都是块级元素，每个标题会单独占用一行的空间。

一级标题

二级标题

四级标题

六级标题

图 2-12　不同级别的标题显示

3. em 元素和 strong 元素

em 和 strong 元素都表示强调(Emphasis)，区别在于后者要比前者的强调程度更深。在大多数浏览器中，em 元素中的文字会以斜体方式显示，而 strong 元素中的文字会以粗体方式显示。



请看示例：

```
<p>在 HTML 文档中，为了使代码清晰，每个元素<em>应该</em>有结束标记，但这不是强制的。而在 XHTML 文档中，每个元素<strong>必须</strong>要有结束标记，这是<em><strong>XHTML 文档规范要求</strong></em>的。</p>
```

如图 2-13 所示为代码的显示效果。可见，em 元素中的内容以斜体显示，strong 元素中的内容以粗体显示，二者一起使用则以粗斜体显示。

在HTML文档中，为了使代码清晰，每个元素*应该*有结束标签，但这不是强制的。而在XHTML文档中，每个元素**必须**要有结束标签，这是***XHTML文档规范要求***的。

图 2-13　显示效果

4. br 元素

网页中的文字会在到达容器边缘时自动产生换行。但有时我们需要在一段文字中的某个特定位置产生换行，br 元素的作用就在于此。br 元素本身没有内容，属于空元素，在 HTML 中只要一个
标记即可，但是 XHTML 要求这样的元素是自我关闭的，因此需要将此标记写成
。请看示例：

```
<h3>如梦令</h3>
<p>
常记溪亭日暮。<br />
沉醉不知归路。<br />
兴尽晚回舟，误入藕花深处。<br />
争渡。<br />
争渡。<br />
惊起一滩鸥鹭。<br />
</p>
```

每个 br 元素会产生一个换行，效果如图 2-14 所示。

如梦令

常记溪亭日暮。
沉醉不知归路。
兴尽晚回舟，误入藕花深处。
争渡。
争渡。
惊起一滩鸥鹭。

图 2-14　br 元素会产生换行

5. hr 元素

hr 元素会在页面中画出一条水平线(Horizontal Rule)，它属于块级元素，水平线会独占一行。从逻辑上讲，hr 元素将两个不同的区域分隔开来。与 br 元素一样，hr 元素也不包含任何内容，因此该元素标记应写成<hr />。不同浏览器对 hr 元素的显示有些差异，这可以通过 CSS 进行调整。请看示例：



```
本行文字位于水平线之上。
<hr />
本行文字位于水平线之下。
```

效果如图 2-15 所示。

本行文字位于水平线之上。

本行文字位于水平线之下。

图 2-15　hr 元素会生成一条水平线

6. img 元素

img 元素用来向页面添加图像，它属于内联元素，元素前后不产生换行。因此插入的图像可以和文本融入在一起。img 的 src 属性指明了图像所在的位置，通常浏览器可以显示 JPEG、GIF、PNG 和 BMP 格式的图像，显示效果由浏览器决定。当网络问题或者浏览器支持问题导致图像无法显示时，img 的 alt 元素可以在图像的位置显示一些文字，以表示该图像的用途。img 元素没有内容，在 XHTML 中可以在<img>标记之后添加一个结束标记</img>，或者在起始标记中的最后添加“/”。请看示例：

```
<p>点击记事本图标<img src="images/notepad.gif" />可以启动记事本程序。</p>
```

如图 2-16 所示为浏览器中的显示效果。



图 2-16　使用 img 元素添加图像

7. div 元素

与前面介绍的元素相比，div 元素的语义性不是很强，div 元素表示一个逻辑上独立的区域(Division)，其作用在于将相关的内容和元素组织在一起。div 元素通常用来将页面中的所有内容组织成一个个相对独立的区域，每个区域的 div 元素拥有唯一的 ID 属性，以便使用 CSS 或者脚本语言对页面进行处理和控制。比如某个网站的顶部含有公司标识和导航菜单，中间有些新闻或公司的介绍，底部是公司的地址以及联系方式，使用 div 元素就可以划分出这三个不同的区域，比如：

```
<!--company logo and navigation-->
<div id="top">
...
</div>
<!--middle content-->
<div id="middle">
...
</div>
<!--address and contacts-->
<div id="footer">
...
</div>
```

div 属于块级元素，前后会有换行，且占满水平方向的空间。div 元素可以包含文本和其他任何类型的元素。由于 div 元素本身没有很强的语义性，因此其内部的文本应当尽量放在有意义的元素中。

div 对于组织页面内容非常有效，但是也不能过分地使用它(一些设计者甚至使用 div 来达到某些视觉上的效果)。记住，div 元素是用来组织内容的，而不是布局页面的工具。在本书的页面布局部分还会讲到如何用 div 元素结合 CSS 对页面进行布局。

8. span 元素

span 元素的作用与 div 元素的作用相同，只不过 span 属于内联元素，通常用来在一段文本中划分不同的区域，有时也配合 CSS 来达到某些视觉上的效果。span 元素也要避免被滥用，当没有合适的强语义性元素时才考虑使用 span 元素。请看下例：

```
电子邮件: <span style="color:red;">huistd@163.com</span>
```

该代码用 span 元素将电子邮件地址划分出来，并通过 style 属性指定其中的文字为红色。

2.4.5 文档树

在讲解文档树这个概念之前，我们先来看一段(X)HTML 代码：

```
<html>
    <head>
        <title>文档树</title>
    </head>
    <body>
        <h1>什么是文档树</h1>
        <p><em>文档树</em>可以帮助我们更好地分析页面结构。</p>
    </body>
</html>
```

这段代码内容非常简单，它以标记<html>起始，以标记</html>结束，中间又包含了 head 和 body 两个元素，head 包含了一个 title 元素，body 包含了一个 h1 和一个 p 元素，而 p 又含有一个 em 元素。

元素之间这种包含与被包含的关系可以看作是一种父子关系，比如 p 元素就是 em 元素的父元素(Parent Element)，反过来 em 元素是 p 元素的子元素(Child Element)，再比如，body 元素是 h1 元素和 p 元素的父元素；title 元素是 head 元素的子元素。

子元素可以称父元素以及包含该父元素的所有元素为祖先(Ancestor)元素，而父元素可以称子元素以及子元素所包含的所有元素为后代(Descendants)元素。比如：html、body 和 p 元素都是 em 元素的祖先元素；p 元素、em 元素都是 body 和 html 元素的后代元素。拥有同一个父元素的多个元素之间称作兄弟(Siblings)元素。比如，head 和 body 就互为兄弟元素，同样，h1 与 p 元素亦为兄弟元素。

有了这些概念之后，我们就可以用一种类似于树形的结构来描述这个文档了，树根就是文档的根元素(Root Element)，其余元素按照父子顺序依次展开，从而形成一个树形结构，我们把这种树形结构称作文档树(Document Tree)。以上代码就可以用如图 2-17 所示的文档树来描述。



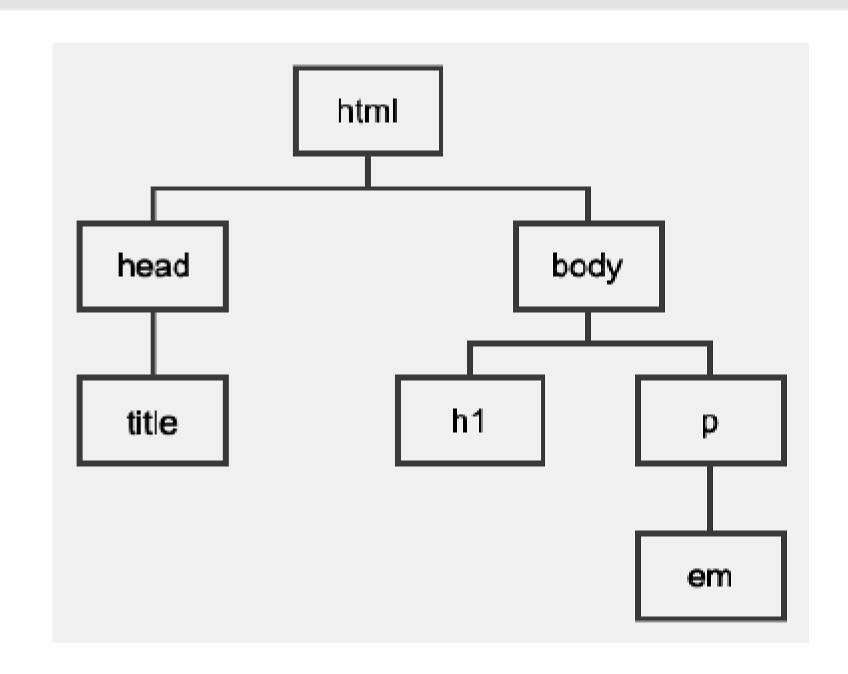


图 2-17　文档树结构

2.5　编写符合标准的(X)HTML

2.5.1　使用语义化标记

在第 1 章中曾经说过，可以使用语义化的(X)HTML 对文档进行结构化，从而构建符合标准的 Web 页面。那么如何创建结构良好的、语义化强的 Web 文档呢？

(X)HTML 中的每一个元素都有特定的含义，在结构化一份 Web 文档时应选用与文档内容相符的元素。比如普通段落用 p 元素表示，标题根据不同级别使用元素 h1~h6，新闻列表应放在 li 元素中等。避免文本直接出现在 body 元素中或者 div 这种语义性不强的元素中。

表 2-2 总结了一些经常被忽略但却非常有用的元素，使用这些元素可以提高页面的结构化程度，使文档内容的用途更加明确。

表 2-2　经常被忽略的(X)HTML 元素

元素名称	元素含义
address	标记一段地址
dl	表明一个定义列表(通常用于表示术语/定义对，也可用来表示其他名/值对)
dt	表明定义列表中的术语部分
dd	表明定义列表中的术语定义部分
code	将文本标记为代码
blockquote	表示一段引用(通常表示内容较多的引用)
q	表示一段引用(通常表示内容较少的引用)
label	给表单元素添加标记
th	标明表格中行和列的表头
thead	表示表格头部区域
tfoot	表示表格底部区域
fieldset	将表单元素成组

续表

元素名称	元素含义
button	创建表单按钮
cite	表示一段文字的出处(通常为文献、书籍、杂志等的标题)
samp	表示程序、脚本中的范例
kbd	表示用户输入的文本
abbr	表示缩略语
acronym	表示只取首字母的缩写词

2.5.2 避免使用具有表现功能的元素和属性

具有表现功能的元素没有任何语义，元素中的内容只有外观上的变化。我们要避免使用这样的元素，而把外观控制权全部交给 CSS。比如，font 元素可以用来指定字体特性，b、i、u 元素可对文本进行加粗、斜体显示和添加下划线。另外，某些属性也可以用来控制外观，比如 align 属性可控制元素的对齐方式，这样的属性我们也要避免使用。

除此之外，我们还要注意不要误用元素。早期的浏览器不支持 CSS，一些设计者使用表格元素安排内容的位置，通过将不同内容放置于各个单元格中来达到复杂的布局效果。如图 2-18 所示为使用表格布局的示意图，我们看到页面中会出现大量嵌套使用的表格。这使得代码可读性大大降低、不利于维护。

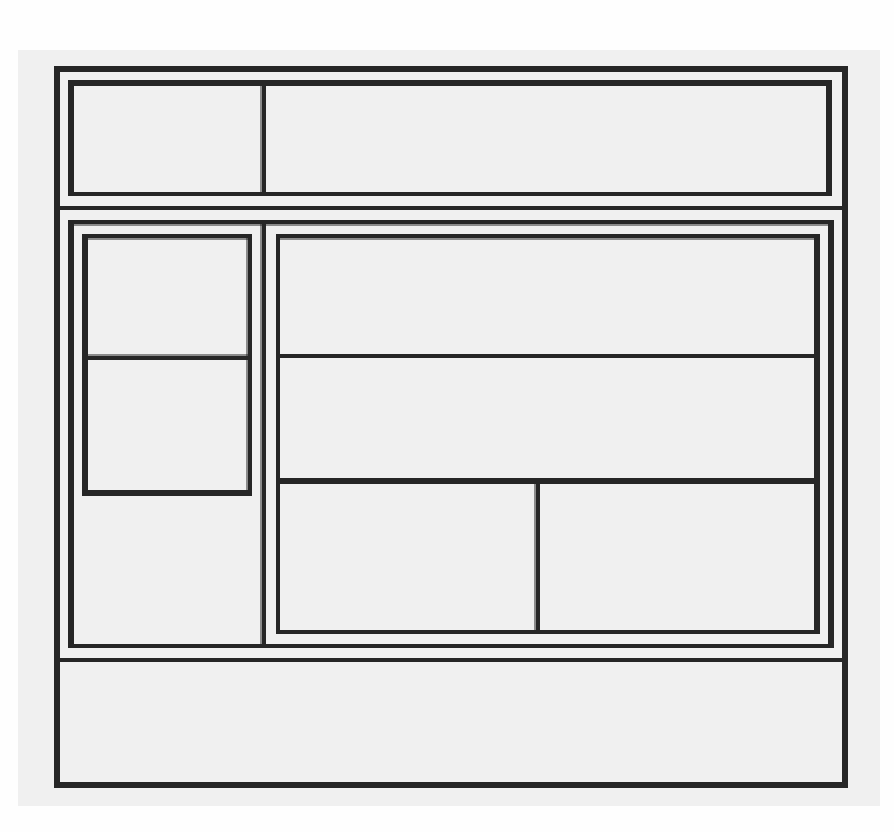


图 2-18　使用 table 进行布局和排版

记住，(X)HTML 的作用是表示页面中的内容是什么，而不是什么样子。

2.6 小　结

本节介绍了有关(X)HTML 的基础知识，为后面学习 CSS 做好准备，读者要想进一步学习(X)HTML，可以参考其他相关的书籍。



(X)HTML 是一种文档标记语言，其目的在于为纯文本文档添加结构和语义。

(X)HTML 文档由一个个元素组成，元素标识着文档内容的含义。元素由起始标记、元素内容和结束标记组成。有些元素不包含内容，称为空元素。元素中可以包含属性，用来提供一些附加信息。属性由属性名称加属性值组成。

(X)HTML 元素可分为两大类：块级元素和内联元素，它们在显示效果上有所区别。

(X)HTML 文档通常包含文档类型声明、文档头和文档体。由于元素可以嵌套出现，所以(X)HTML 文档可以使用一种树型结构来表示，这就是文档树。

与 HTML 相比，XHTML 的语法要求更为严格，比如标记、属性名等必须小写；属性值必须用双引号括起来；必须有结束标记等。

应当使用(X)HTML 元素来结构化文本信息，不要使用只有表现功能的元素和属性，表现的控制应由 CSS 来完成。

在下一章，我们将介绍有关 CSS 的一些基本概念。

第 3 章　CSS 的基本概念

本章将向读者介绍有关层叠样式表的基本概念以及它在 Web 设计中所起的作用、CSS 规范的不同版本之间有哪些区别、如何将 CSS 应用到 Web 页面中、如何管理 CSS。接下来我们会编写本书中的第一个 CSS 样式，然后详细介绍 CSS 的语法规则和代码的编写风格。

良好的开发和调试工具对于 Web 开发人员来说是必不可少的，它直接影响着开发效率。本章在最后将介绍几种不同类型的开发工具，从最简单的文本编辑器到功能强大的 Web 开发集成环境，接着将介绍一些常用的调试工具，它们可以帮助开发人员迅速查看页面结构，寻找问题的根源，从而加快开发的速度。

本章主要内容

- 什么是层叠样式表、样式和层叠的含义是什么
- CSS 能为 Web 设计做些什么
- CSS 规范版本的发展过程
- 如何将 CSS 应用到(X)HTML 页面中
- 如何组织好 CSS 代码
- CSS 样式的基本组成
- 如何添加注释信息
- 代码风格
- 常用的 CSS 开发工具
- 常用的 CSS 调试工具

3.1　什么是 CSS

CSS 是英文 Cascading Style Sheet 缩写形式，中文译为层叠样式表或级联样式表。Web 设计者可利用它来定义文档的样式，这里指的文档不仅限于(X)HTML。通过 CSS，设计者可控制文档的字体、颜色、图像、表格、链接和布局格式，同时设计者也可以将表示样式外观的信息从内容中分离出来，集中放置在页面的某一部分，甚至可保存为独立的文件，从而减少文件的大小、节省网络的带宽、节约 Web 设计者维护代码的时间。CSS 有如此多的好处，掌握和使用好它对于 Web 设计者来说是非常必要的。

3.1.1　何为样式

样式一词对我们来说并不陌生，即使尚未接触 CSS 的人也不难理解样式的含义。当你使用 Microsoft Word 一类的字处理程序时，几乎总要更改某些样式以达到更好的显示效果，比如设定标题为加粗的三号黑体字，每一段的开始留出两个空格等。样式表不能孤立地使用，因为

它本身并不包含任何内容信息。当然 CSS 也不仅仅能同 Web 文档一起使用，它还能定义 XML 甚至软件界面的样式。

3.1.2　何为层叠

与样式相比，了解层叠一词的 CSS 初学者可能就比较少了，层叠是 CSS 中的术语，它包含了一系列的规则，浏览器根据这个规则来确定样式应该如何应用到页面的各个元素中去。本书第 6 章将会详细介绍层叠概念。

3.2　CSS 的作用

3.2.1　排版与风格设计

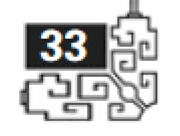
(X)HTML 只是一种简单的结构化标记语言，还不能够很灵活地控制页面的外观。而 CSS 的引入，正好弥补了这一不足。通过 CSS，Web 设计者能够精确而灵活地对页面进行排版，设计出精美的效果。

使用 CSS 设计出来的网页究竟是什么样子的呢？我们从首届 CSS 世界大赛(The first CSS World Awards)选出一部分获奖作品供大家欣赏，让读者领略一下 CSS 的魅力。

(1)　年度最佳站点(site of the year)：UX Magazine(http://www.uxmag.com/)，如图 3-1 所示。



图 3-1　首届 CSS 世界大赛年度最佳站点(UX Magzine)



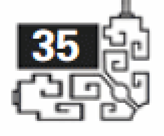
(2) 机构类第三名：Reductostart(http://www.reductostart.ro/)，如图 3-2 所示。

图 3-2　首届 CSS 世界大赛机构类第三名(Reductostart)

(3) 商业类第一名：The City Church(http://www.thecity.org/)，如图 3-3 所示。

当然，要想设计出如此绚丽的页面，除了掌握好 CSS，能编写出符合规范，健壮性高的样式代码外，还需要有良好的美术功底和审美观，最重要的是有个好的创意。

图 3-3　首届 CSS 世界大赛商业类第一名(The City Church)



3.2.2 简化的 Web 开发

在 CSS 出现之前，Web 设计者全部使用(X)HTML 元素来控制页面样式，比如用 font 元素来设置文字的颜色、大小等：

```
<font size="+2" face="Arial" color="red">使用 font 元素控制文字样式。</font>
```

上面这段代码将字号设置为+2，字体为 Arial，颜色为红色，尽管这样做确实可以达到修改样式的目的，但如果整个页面甚至整个网站上所有文字的样式都按照此方法来设置，不仅页面会大量存在<font>标记，增加了文件大小，而且修改和维护工作的繁重程度也是可想而知的。

CSS 的出现消除了这种设计方式，大大减轻了代码的维护工作。我们可以将文字放在表示段落的标记<p>内：

```
<p>使用 CSS 控制文字样式。</p>
```

然后将如下代码放置在(X)HTML 页面的 head 元素中：

```
<style type="text/css">
p {
    font-size:x-large;
    font-family:Arial;
    color:red;
}
</style>
```

现在页面内容和样式就彻底分离开了，二者互不干扰，这里样式信息被集中放置在页面的首部，浏览和修改起来都很方便。后面我们还会讲到样式信息可以单独保存到独立的文件中，这样就可以完全独立地添加、编辑样式，而不会影响包含内容的文件。

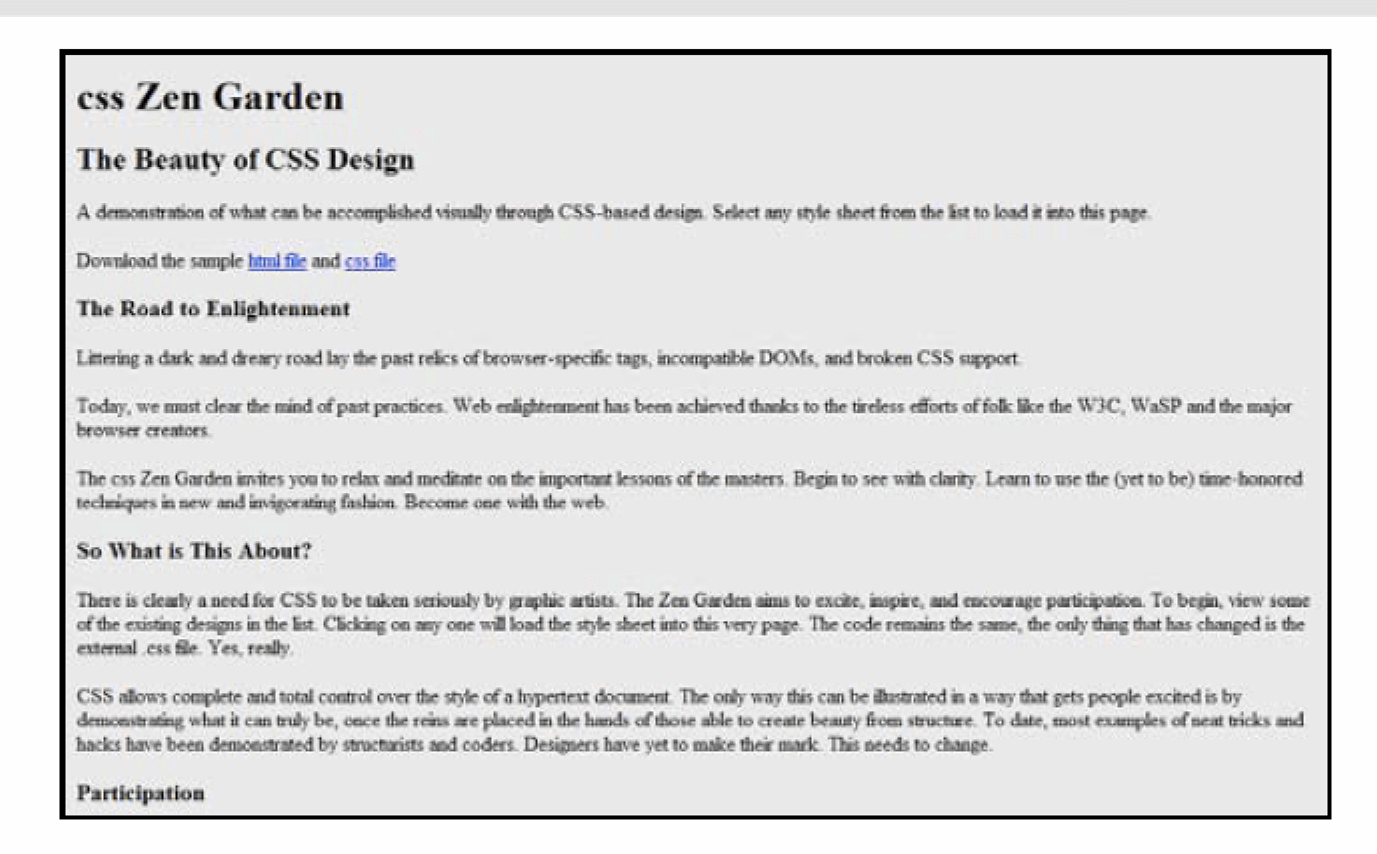
著名的“CSS 禅意花园”网站(http://www.csszengarden.com)就通过为同一个 XHTML 文档指定不同的样式表文件的方式，实现了数以百计风格各异的页面效果。图 3-4 展示了两位不同设计师的作品，而图 3-5 则是除去 CSS 样式后的纯 XHTML 页面。





图 3-4 两个风格迥异的“CSS 禅意花园”，请相信它们使用的是相同的 XHTML 文档





css Zen Garden

The Beauty of CSS Design

A demonstration of what can be accomplished visually through CSS-based design. Select any style sheet from the list to load it into this page.

Download the sample html file and css file

The Road to Enlightenment

Littering a dark and dreary road lay the past relics of browser-specific tags, incompatible DOMs, and broken CSS support.

Today, we must clear the mind of past practices. Web enlightenment has been achieved thanks to the tireless efforts of folk like the W3C, WaSP and the major browser creators.

The css Zen Garden invites you to relax and meditate on the important lessons of the masters. Begin to see with clarity. Learn to use the (yet to be) time-honored techniques in new and invigorating fashion. Become one with the web.

So What is This About?

There is clearly a need for CSS to be taken seriously by graphic artists. The Zen Garden aims to excite, inspire, and encourage participation. To begin, view some of the existing designs in the list. Clicking on any one will load the style sheet into this very page. The code remains the same, the only thing that has changed is the external .css file. Yes, really.

CSS allows complete and total control over the style of a hypertext document. The only way this can be illustrated in a way that gets people excited is by demonstrating what it can truly be, once the reins are placed in the hands of those able to create beauty from structure. To date, most examples of neat tricks and hacks have been demonstrated by structurists and coders. Designers have yet to make their mark. This needs to change.

Participation

图 3-5　除去 CSS 之后的纯 XHTML 页面

图 3-6 展示了 XHTML 和 CSS 文件的关系，针对同一个 XHTML 文件应用不同的 CSS 样式，仿佛为网页穿上了不同的衣服一样，使其能够产生截然不同的外观风格。



图 3-6　针对同一个 XHTML 文件使用不同的 CSS 样式，页面将产生不同的效果

3.3　CSS 的起源及发展

CSS 是在万维网联盟(W3C，http://www.w3.org)的推动下，由 Web 开发人员与浏览器的开发者共同协作完成的。W3C 发布的规范(Specification)称作建议(Recommendation)。W3C 徽标如图 3-7 所示。



图 3-7　国际万维网联盟 W3C 徽标



CSS 规范是由 W3C 的层叠样式表工作组发布的。目前，该工作组已发布了两套完整的 CSS 建议：CSS1 和 CSS2。CSS2 的升级版本 CSS 2.1 的草案已经制定，其成为正式建议也指日可待。CSS3 还处于草拟阶段，它将包含更多的内容。

1. CSS1

层叠样式表级别 1(CSS Level 1，简称 CSS1，http://www.w3.org/TR/REC-CSS1)规范作为 W3C 的建议于 1996 年 12 月 17 日发布，并于 1999 年 1 月 11 日发布修订版。

CSS1 定义了各种各样的样式(比如字体、颜色等)，并将其应用到(X)HTML 文档中。但是与 CSS2 相比，CSS1 提供的样式和功能还是非常有限的。

2. CSS2

CSS2(CSS Level 2)发布于 1998 年 5 月 12 日(http://www.w3.org/TR/REC-CSS2)，它在 CSS1 的基础上扩展了许多功能，比如增加了新的选择符、伪类和伪元素，改进了文本和字体的属性等。CSS2 使得 Web 设计者不仅能为(X)HTML 创建样式，还能为其他类型的结构化文档(比如 XML)和特定设备(比如打印机、语音设备)创建样式。CSS2 有效地区分了文档内容和其表现，简化了 Web 开发过程。

3. CSS 2.1

W3C 于 2002 年 8 月 2 日发布了 CSS 2.1(CSS Level 2 Revision 1，http://www.w3.org/TR/CSS21)工作草案(Working Draft)。目前它已经成为候选建议(Candidate Recommendation)，发布日期为 2007 年 7 月 19 日。CSS 2.1 修正了 CSS2 中存在的几处错误，增加了一些得到广泛支持的属性。

尽管 CSS 2.1 目前尚未成为 W3C 正式建议，但已经成为事实上的标准。大多数浏览器都能支持 CSS 2.1 版本的样式，大多数 Web 设计者也都按照 CSS 2.1 编写样式。因此，本书内容也以 CSS 2.1 为依据。

4. CSS3

与此前的 CSS 规范不同，CSS3(http://www.w3.org/TR/css3-roadmap)规范不再由单一的文档发布，而是按照不同模块独立发布的。这样有利于理清各部分之间的联系，减少整个文档的大小。各个模块可独立进行测试和更新，使用起来更为灵活。CSS3 将支持国际性语言，支持使用可下载的字体，支持和 SVG、MathML 和 SMIL 的整合。CSS3 还会通过 CSS 行为扩展(Behavioral Extensions to CSS)实现(X)HTML、样式表和脚本语言之间的进一步整合。

目前 CSS3 的一部分模块的规范还处于工作草案阶段，另一部分已经成为候选建议。

3.4 应用到 Web 页面

我们已经介绍了(X)HTML 和 CSS 的一些基本概念，那么怎么将 CSS 应用到(X)HTML 页面当中呢？这就是本节所要解决的问题。将 CSS 应用到 Web 页面的方式有很多种，至于选用哪一种，要根据具体情况而定。



3.4.1　内联样式

内联样式(Inline Style)通过(X)HTML 元素的 style 属性为元素直接添加样式信息。style 属性的内容是由 CSS 属性加属性值组成，例如：

```
<p style="color:red;">这段文字显示为红色。</p>
```

将 p 元素内的文字颜色设为红色。style 属性是(X)HTML 的核心属性之一，每一个(X)HTML 元素都支持使用 style 属性。当你想快速验证某些样式的效果时，使用内联样式会很方便。但这种方式也存在很多问题。CSS 的初衷是要将页面的内容和表现进行有效的分离，但使用内联样式则会违背这一原则。假如页面中多个 p 元素都要求使用红色文字，这就需要在每个 p 元素的 style 属性中声明“color:red”，这将导致页面中到处都存在着重复的样式定义，维护起来非常不便，因此在实际开发中很少使用。

3.4.2　嵌入样式

嵌入样式(Embedded Style)的内容仍然和(X)HTML 内容写在一起，但是与内联样式不同，嵌入样式的所有样式属性都被包含在一个单独的 style 元素中，该元素位于(X)HTML 的 head 元素中。请看下面这段代码：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>嵌入样式示例</title>
<style type="text/css">
p{
    color:red;
}
</style>
</head>
<body>
<p>这段文字显示为红色。</p>
</body>
</html>
```

把这段代码保存为 embedded.html 并在浏览器中打开，可以看到段落内的文字显示为红色。

嵌入样式的使用频率要比内联样式要高，这种方式使得样式信息能够集中在一起，在一定程度上减少了冗余的样式信息，方便代码的修改和维护。嵌入样式固然要嵌入到(X)HTML 页面中，所以每次载入(X)HTML 时样式信息也要跟着载入。另外，若多个页面的样式相同，也需要重复地在每个页面都包含一段相同的样式代码，代码冗余问题还是存在。由于每个页面都含有样式信息，维护代码的工作量也会很大。

3.4.3 外部样式

外部样式(External Style)实现了样式信息和内容信息的完全分离，CSS 样式作为独立的文件存在，多个页面可以使用同一个样式表文件。请看下面的示例。

首先建立 external.css 文件，文件内容如下：

```
p {color:red;}
```

将该文件和下面的 XHTML 文档放在同一目录下，XHTML 代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>嵌入样式示例</title>
<link rel="stylesheet" type="text/css" href="external.css" />
</head>
<body>
<p>这段文字显示为红色。</p>
</body>
</html>
```

把它保存为 exteranl.html，并在浏览器中打开，文字颜色仍是红色。这段(X)HTML 代码不再包含任何样式信息了，它只负责放置内容，样式信息都在外部样式文件中。使用(X)HTML 代码<link rel="stylesheet" type="text/css" href="external.css" />将样式表 external.css 与当前文档关联起来，其中 rel 属性为 stylesheet，表明外部文件是该文件的样式表，type 为 text/css，表明是 CSS 类型，href 指定了外部样式表文件的位置。通过这种方式，可以将该样式表应用到多个(X)HTML 页面，而样式信息是唯一的，这就是外部样式的最大优势。同时，一个页面也可以添加多个 link 元素来引用多个样式表文件。另外，一旦浏览器下载了样式表文件，会把它们缓存起来而不必重新下载，因而可以节省带宽，加快网页的载入速度。

延伸： 当 link 元素的 rel 属性为“alternate stylesheet”时，该元素所指向的样式表文件可作为页面的一个可选样式，默认情况下浏览器不会应用该样式。在 Firefox、Opera 等浏览器中，可以通过菜单中的相关选项选择这些可选样式，样式的名称由 link 元素的 title 属性确定。遗憾的是，IE 浏览器没有提供此功能，因此只能依靠脚本代码等其他方式进行选择。

到此为止，我们学习了三种将样式表应用到(X)HTML 页面的方式，它们是内联样式、嵌入样式和外部样式，其中外部样式的使用频率较高，它充分地实现了样式和内容的分离。若样式具有页面唯一性，则也可以使用嵌入样式。若想快速验证某些样式的效果，临时使用内联样式也是可以的。还有一种方式可以将外部样式表文件导入进来，这种方式是 CSS 自有的功能，我们在后续章节中进行介绍。



3.5　管理 CSS

前一节介绍了 3 种将 CSS 应用到 Web 页面的方法，实际上只有外部样式才能充分发挥 CSS 的优势。一般的大中型网站会包含成千上万个页面，使用外部样式就可以通过一个或几个 CSS 文件为整个网站添加样式。下面我们就来讨论一下如何组织和管理这些 CSS 文件。

最简单的组织方式就是所有页面共用一个 CSS 文件，这也仅仅适用于页面数量较少、页面样式单一的网站。假如页面数量过多且样式布局各异，此时的样式表文件可能会包含上千行的代码，文件也会随之增大。这样的样式表对于开发人员来说是难以维护的，同时，即使用户浏览一个页面也要下载这个“庞大”的 CSS 文件，影响了页面加载速度，也会增加网络流量。

为了避免使用单一样式表所带来的负面影响，我们可以将 CSS 代码模块化，分别存放在不同的样式表文件中。一种可行的方案是按照 CSS 的用途进行划分，比如将控制页面布局结构的代码存放在 layout.css 中，将控制文本格式的代码存放在 text.css 中。这样可以有效提高代码的可维护性，但还是不能减少访问者的下载量。另一种方案是按照 CSS 所属的页面进行划分，比如首页采用 index.css，产品介绍页面使用 product.css。这样，样式表文档相互独立，能够有效地加快页面的下载速度。

3.6　编写第一个 CSS 样式

3.6.1　增加样式

我们使用第 2 章编写的第一个 HTML 文档，为它增加文档类型声明，使其转换成为一个符合 XHTML 1.0 过渡型规范的 XHTML 文档。在文档头部区域添加一个 link 元素，引用一个名为 first.css 的外部 CSS 文件。修改后的文件内容如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<link type="text/css" rel="stylesheet" href="first.css" />
<title>第一个 Web 页面</title>
</head>

<body>
<h1>Hello, XHTML and CSS!</h1>
</body>
</html>
```

把它保存为 first-html-with-css.html。

再次打开编辑器，在新建的文档中输入以下内容：

```
body{
    background-color:gray;
    text-align:center;
}
h1{
    font-family:"Times New Roman", Times, serif;
    font-size:30px;
    color:white;
}
```

把它保存为 first.css，注意 first.css 和前面的 first-html-with-css.html 要处于同一目录中。用浏览器打开 first-html-with-css.html 文档，会产生如图 3-8 所示的效果。

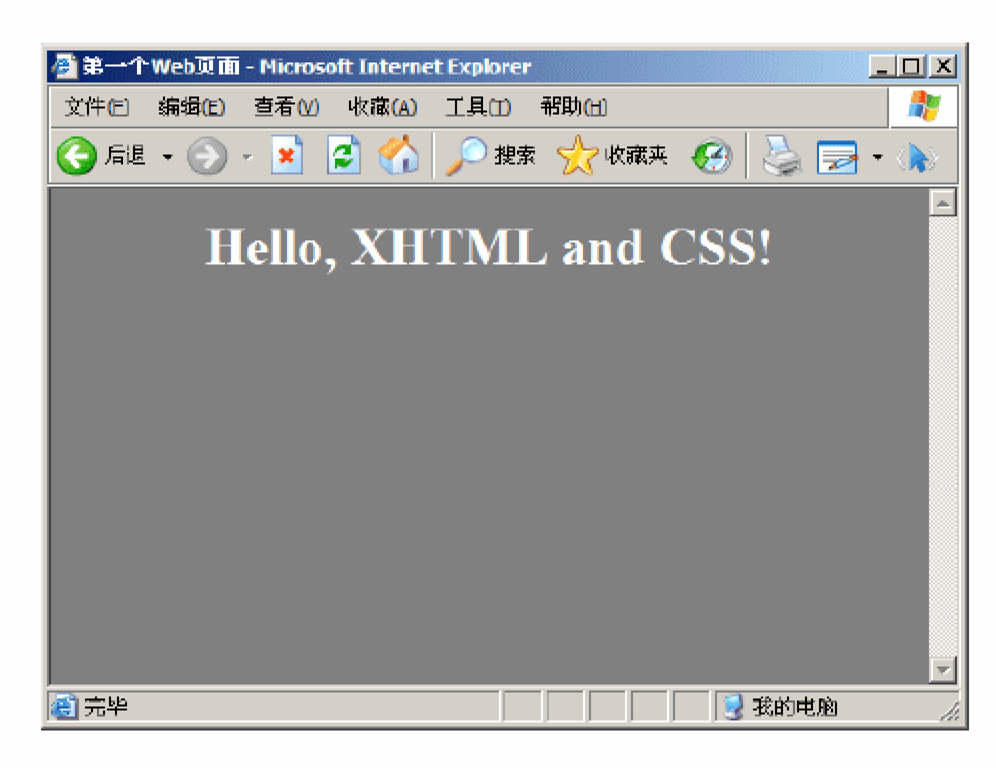


图 3-8　增加 CSS 样式

除了在原有的 HTML 文档之中增加一个文档类型声明和一个 link 元素之外，文档主体区域没有增加任何内容。外部的 CSS 文档使得页面的显示效果与第 2 章的图 2-2 有了区别。对于 first.css 文件中各部分的含义，读者可暂时不必理会，这里只是让读者了解一下 CSS 样式的编写过程，以及如何应用到一个 Web 页面中。

3.6.2　本书约定

以下几点是本书的一些约定，请读者在阅读本书和实际练习中注意：

- 除非做特殊说明，本书出现的 CSS 代码均符合 CSS 2.1 规范。
- 本书示例全部采用 XHTML。限于篇幅，大部分示例只给出 body 元素之中的代码片段，我们省去了文档类型声明、文档头部和<body>标记。读者在实际编写时应该使用完整的 XHTML 代码结构，并添加 XHTML 1.0 过渡型的文档类型声明，设置文档字符集为 gb2312，即简体中文(这些都是本书示例实际所采用的)。代码中的重复部分将以省略号表示。当然，示例也同样适用于 HTML。
- 本书所列举的CSS 代码片段均可放置在位于 XHTML 文档头部的 style 元素中，且 style 元素需要添加属性 type="text/css"。也可将其单独保存为样式文档，并用 link 元素进行引用。建议读者在学习过程中采用第一种方式。



- 除非做特殊说明，本书所有示例可以保证对 Windows 平台下的 IE6、IE7 和 Firefox2 浏览器的兼容性，并使用 IE6 演示页面效果。
- 除非做特殊说明，本书不区分 HTML 和 XHTML，并统称为(X)HTML。

例如书中出现以下代码(取自第 12 章，12.2.2 小节)：

```
ul{
    list-style-image:url(images/list.gif);
}

<h3>CSS 中列表样式属性</h3>
<ul>
    <li>list-style-type</li>
    <li>list-style-image</li>
    <li>list-style-position</li>
    <li>list-style</li>
</ul>
```

则它的完整形式为：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>Untitled Document</title>
<style type="text/css">
ul{
    list-style-image:url(images/list.gif);
}
</style>
</head>

<body>
<h3>CSS 中列表样式属性</h3>
<ul>
    <li>list-style-type</li>
    <li>list-style-image</li>
    <li>list-style-position</li>
    <li>list-style</li>
</ul>
</body>
</html>
```

3.7　CSS 规则详解

CSS 规则组成了样式表的基本内容，每一条规则定义了一系列样式以及该样式的使用



方式。

CSS 规则由一个选择符(Selector)和一段声明块(Declaration Block)组成，声明部分用一对大括号包括起来，中间包含 0 个或多个声明(Declarations)，每个声明之间用分号“;”隔开。每一条声明是由一个属性(Property)加属性值(Value)组成，二者用冒号“:”隔开。

一般来说，一条 CSS 规则会有如下形式：

```
选择符 {属性:值; 属性:值; ...}
```

其中，选择符表明该条规则应用到页面的哪个部分。选择符的内容可以是(X)HTML 元素名称，这是形式最简单的选择符，如果需要进行更精确的选择，就要在选择符中增添附加条件。CSS 规范中定义了多种类型的选择符，但是不同浏览器对它的支持也不尽相同，我们在下一章将会详细地介绍。

属性名是由 CSS 规范定义的，不同的属性定义了丰富多彩的页面样式和效果，比如背景颜色、高度、字体等。属性值是一个关键字，或者是由空白分隔的多个关键字，它定义了如何设置属性，比如把背景色设为绿色、把某个 div 元素的宽度设为 100 个像素、把正文文字设为宋体等。

下面定义一条非常简单的规则：

```
h1 {color:blue;}
```

它的各部分含义如图 3-9 所示。h1 是选择符，它选择页面中所有的 h1 元素，大括号之间的声明部分只包含一条声明语句，其中属性名称是 color，表示文字颜色，属性值是 blue，即让文字颜色变为蓝色。整条规则的含义就是让(X)HTML 中<h1>标记内的文字显示为蓝色。

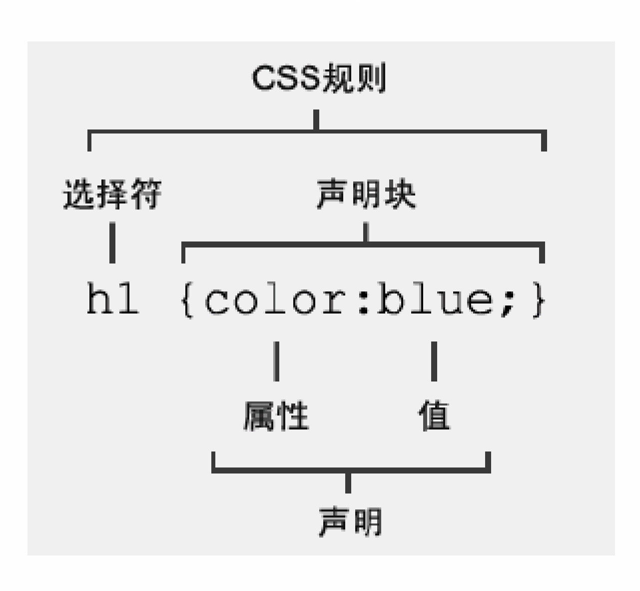


图 3-9　CSS 规则示意图

3.8　注释和风格

3.8.1　代码注释

如果你是一位程序开发人员，那么注释对于你来说最熟悉不过了。与其他语言一样，CSS 允许开发者在代码中添加注释。浏览器在解释 CSS 的过程中会完全忽略掉注释内容。注释作为一种在代码中添加说明的方式，提高了代码的可读性和易维护性，有利于其他开发人员了解样式定义的用意。



CSS 注释以“/*”开始，以“*/”结束。注释可出现在任何地方，但注释不能嵌套使用。

大中型网站的 CSS 样式定义往往是异常复杂的，添加适当的注释信息可以增强代码的可读性，方便其他开发者进行维护。CSS 中常见的注释方式有以下几种。

1. 位于文件首部的注释信息

位于文件首部的注释信息可以包含文档的创建日期，作者，项目信息，附加说明等。当然，如果一个文件中包含了多个人编写的代码，那么也可以在代码中间的分隔处添加类似的信息。下面是一个文件头部注释的例子：

```
/************************************************************
 *  Created date:   22 Sept 2007
 *  Author:         Jia Zheng
 *  Project:        Layout (Hui Blog)
 *  Notes:
 *
 *  Copyright (c) Hui Studio 2006. All Rights Reserved.
 *  Not to be reused without permission.
 *  http://www.huistd.com
 ************************************************************/
```

2. 位于 CSS 规则前的注释信息

在编写样式表时，通常需要大量代码来实现页面某一局部的样式，比如导航菜单。那么可以在控制导航菜单的一系列样式规则前加上一段说明信息，表示下面样式的具体用途。说明信息的语言依开发团队的约定和习惯而定。比如：

```
/* Styles for top menu */
#top menu {
    font-family: Tahoma, sans-serif;
    color: white;
    background-color: black
}
#top_menu a {
    font-size: 10px;
    color: #003399;
    line-height: .82em;
    text-decoration: none;
}
...
```

有时为了使代码结构更加清晰，也可在一系列的样式末尾添加一个表示结束的说明，比如：

```
/* End of styles for top menu */
```

3. 位于声明之后的注释信息

由于 CSS 的样式声明本身就具有良好的自说明性，所以大多数情况下不用写任何注释就可以看懂声明的含义。但有时，声明内容不容易反映出开发者的意图，或者开发者要强调某些



内容，这时就需要在声明后添加注释。

由于不同的浏览器甚至同一浏览器的不同版本对 CSS 的支持程度存在差异，所以有些样式声明可能只在某一特定版本的浏览器中有效果。这时在声明之后加上一条注释可以说明设计者的意图。下面这段 CSS 规则的最后一条声明就被标注了这样的注释：

```
p.newest{
    padding:0;
    margin:0;
    float:left;
    border:1px solid #c2c2d5;
    line-height:20px;
    width:158px;
    height:125px;
    opacity:0.8;    /* for Firefox */
}
```

一些计算机用户会更改系统外观，使得浏览器默认的背景颜色不再是白色。为了提高页面显示效果的健壮性，设计人员会在 body 元素里加上一句设置背景色为白色的声明。在其他人看来这条声明可能是多余的，他们没有意识到用户系统外观更改的问题，样式编写者可以添加一条注释来说明这个问题。

4. 用于调试的注释

在调试样式表时，你可能想要暂时取消某个声明，然后看看页面效果。可以用注释符号将其括起来，浏览器会忽略注释内容，该声明当然也就不会应用到当前页面上，若想恢复，只需去掉注释即可。这是开发样式表的过程中很常用的技巧。

3.8.2 代码风格

设计 Web 页面可以说是一门艺术，编写 CSS 样式也不例外，除了保证样式代码的正确性外，良好的代码风格也是很有必要的。为了使 CSS 代码便于阅读和维护，可以适当添加一些空格、换行或缩进。与(X)HTML 一样，浏览器在解释 CSS 的时候也会忽略其中的空白，包括空格符、换行符和制表符(Tab 键)。

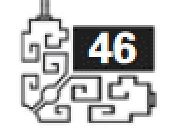
一种常见的风格是将一条规则的内容全部写在一行，例如：

```
body {margin:0;padding:0;font-size:12px;}
```

该规则中出现了三条声明语句，全部写在一行中。

另一种常见的风格是每条声明占用一行，并采用缩进形式，左括号和选择符在同一行，右括号单占一行。例如上面的规则可以写成如下风格：

```
body{
    margin:0;
    padding:0;
    font-size:12px;
}
```



本书中大部分的示例会采用第二种代码风格。另外，规则之间也可以适当添加空行，使代码更容易阅读。结合自己的习惯，以上这两种风格都可以使用。你可以确定自己的风格，一旦确定下来，就该使整个项目的代码风格保持一致。

3.9　工具的重要性

《论语》里提到“工欲善其事，必先利其器”，就是说要做好工作，先要使工具锋利。开发 CSS 样式表文档也是如此，一款良好的开发工具可以帮助 Web 设计人员提高编写样式代码的速度，增加效率。再配合一些辅助工具，可以达到事半功倍的效果。

3.9.1　开发工具

CSS 文档是普通的文本文档，设计人员可以使用多种程序来创建样式表，从简单的文本编辑器到各种专门的开发设计工具。只要能创建和编辑文本文档的程序就能用来编写样式表。

目前也有一些专门的软件工具，它们提供了一些更为高级的功能，能实现可视化设计。文本编辑器、CSS 编辑器和 Web 开发工具是常见的三类开发工具，建议初学 CSS 的读者从简单的工具开始，使用文本编辑器编写代码，这样能帮助你尽快掌握 CSS 的全部基础知识。

1. 文本编辑器

CSS 文档是基本的文本文档，文本编辑器只生成不含任何格式信息的纯文本信息，因此可使用文本编辑器创建和编辑 CSS 文档。文本编辑器很容易找到，每种操作系统都会附带基本的文本编辑器。常见的文本编辑器包括 Windows 的记事本、Linux 或 Unix 的 vi 或 emacs、其他一些文本编辑器(UltraEdit、EditPlus)。如图 3-10 所示为使用 Windows 的记事本编辑样式表。

global.css - 记事本
文件(F)　编辑(E)　格式(O)　查看(V)　帮助(H)

```
#top_dec{
        height:12px;
}
#top_dec_yellow{
        height:10px;
        width:200px;
        float:left;
        background-image:url(../images/top_yellow.gif);
        background-repeat: no-repeat;
        background-position: left top;
}
#top_dec_blue{
        height:10px;
        width:578px;
        float:right;
        background-image:url(../images/top_blue.gif);
        background-repeat: no-repeat;
        background-position: left top;
}
/* 顶部图片 */
#top_img{
        height:125px;
}
/* 滚动图片 */
#rolling_img{
        height:100px;
        background-color:#D6D6D6;
        clear:left;
        margin:1px 0;
        padding:0;
```

Ln 47, Col 2

图 3-10　使用 Windows 的记事本编辑样式表文档

注意： 可以使用 Windows 的写字板或 Microsoft Word 之类的字处理程序来编辑 CSS，但切记要将文件保存为不附带任何特殊格式的文本文档。

2. CSS 编辑器

CSS 编辑器是专为编辑样式表而开发的软件工具。这些工具具有一些高级特性，比如彩色显示样式规则、属性/语法检查、可视化设计等。建议专业的样式表设计人员使用这类工具，它们可以减少编写代码的工作，帮助提高开发效率。

常见的 CSS 编辑器列举如下。

- TopStyle，网址：http://www.newsgator.com/Individuals/TopStyle/Default.aspx
- Rapid CSS Editor，网址：http://www.blumentals.net/rapidcss/
- JustStyle CSS Editor，网址：http://www.ucware.com/juststyle/
- Style Master，网址：http://www.westciv.com/style_master/

如图 3-11 所示为使用 TopStyle 编辑样式表，它可帮助样式表设计人员编写符合包括 CSS2 等标准的样式表。其 CSS 定义选择功能可以选取特定的浏览器或 CSS 规范，此外它还包括样式表检查器、彩色标识代码，以及样式预览等功能。

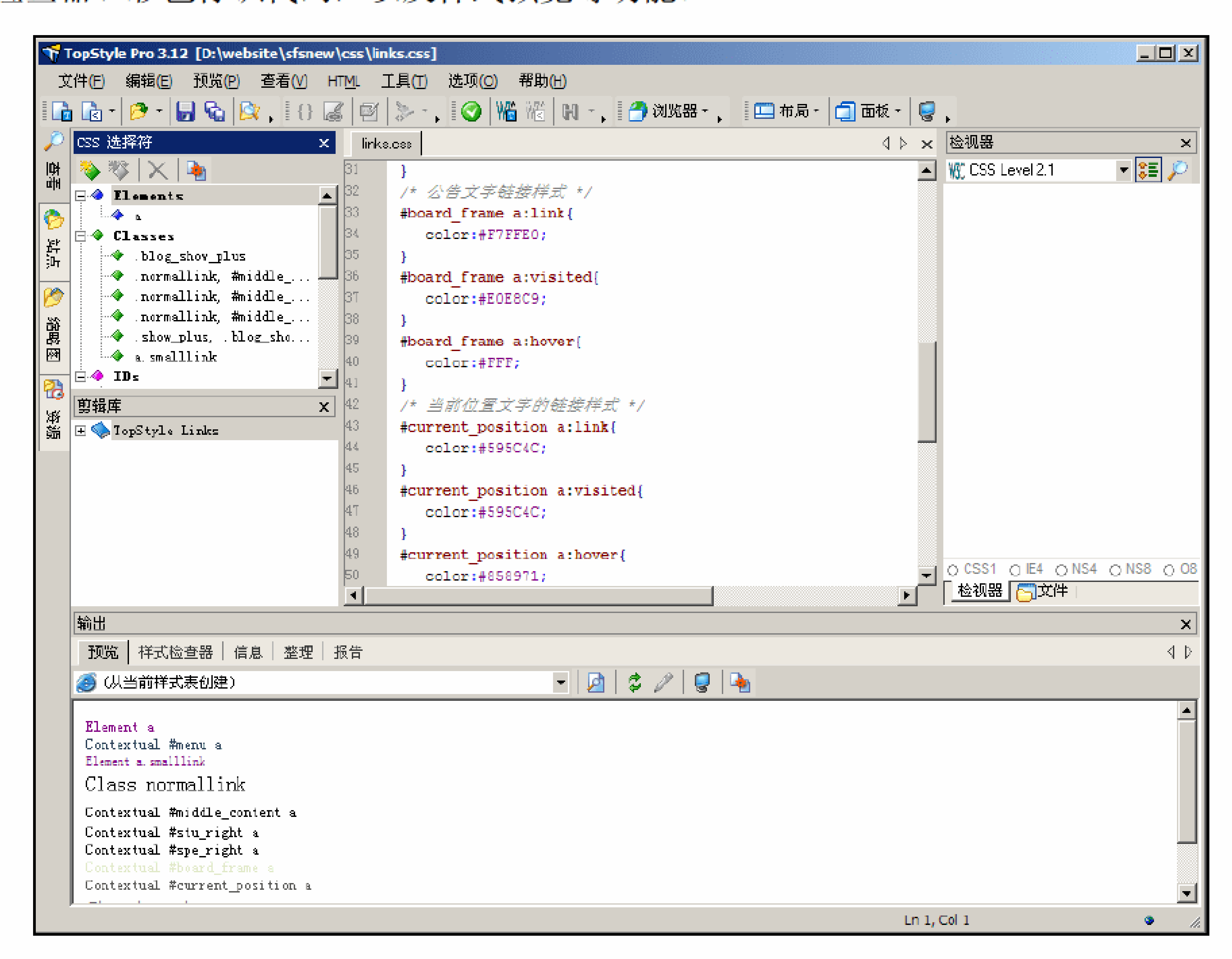


图 3-11 使用 TopStyle 编辑样式表文档

3. Web 开发工具

CSS 和 Web 设计是密不可分的，因此一般大型的 Web 开发工具也会带有 CSS 编辑功能。一些可视化编辑器还可为设计者自动创建样式表，或者只需要输入相应的属性值即可，而源代码编辑器也能让设计者直接编辑样式表。



常见的 Web 开发工具包括：

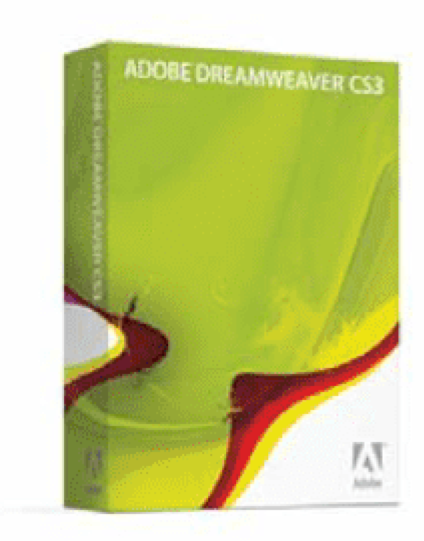
- Adobe 的 Dreamweaver。网址：

  http://www.adobe.com/cn/products/dreamweaver/

- Microsoft 的 Expression Web。网址：

  http://www.microsoft.com/china/expression/expression-web/default.mspx/

Dreamweaver(见图 3-12)可谓是一款功能强大 Web 开发工具，可快速、轻松地完成设计、开发和维护网站和 Web 应用程序的工作。Dreamweaver 可以与 Adobe 家族的其他产品完美地配合起来使用。



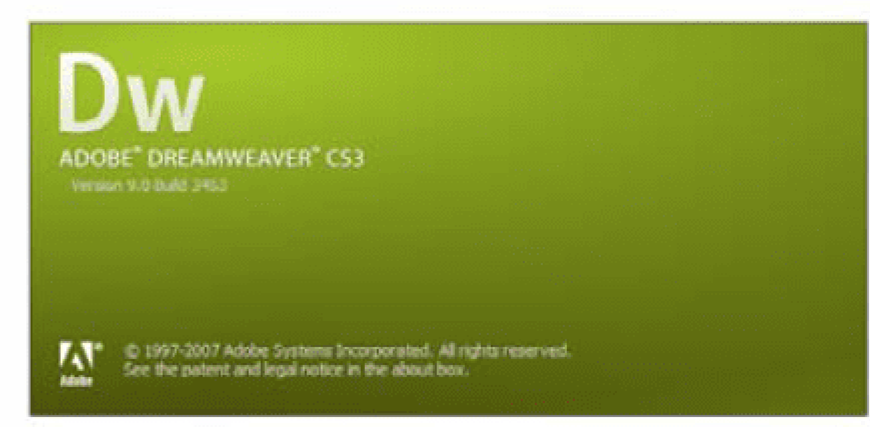


图 3-12　Adobe Dreamweaver CS3 产品

Dreamweaver 是为设计人员和开发人员而构建的，既可以在直观的可视布局界面中工作，也可以在简化的编码环境中工作。如图 3-13 所示为使用 Dreamweaver CS3 编辑样式表文件。

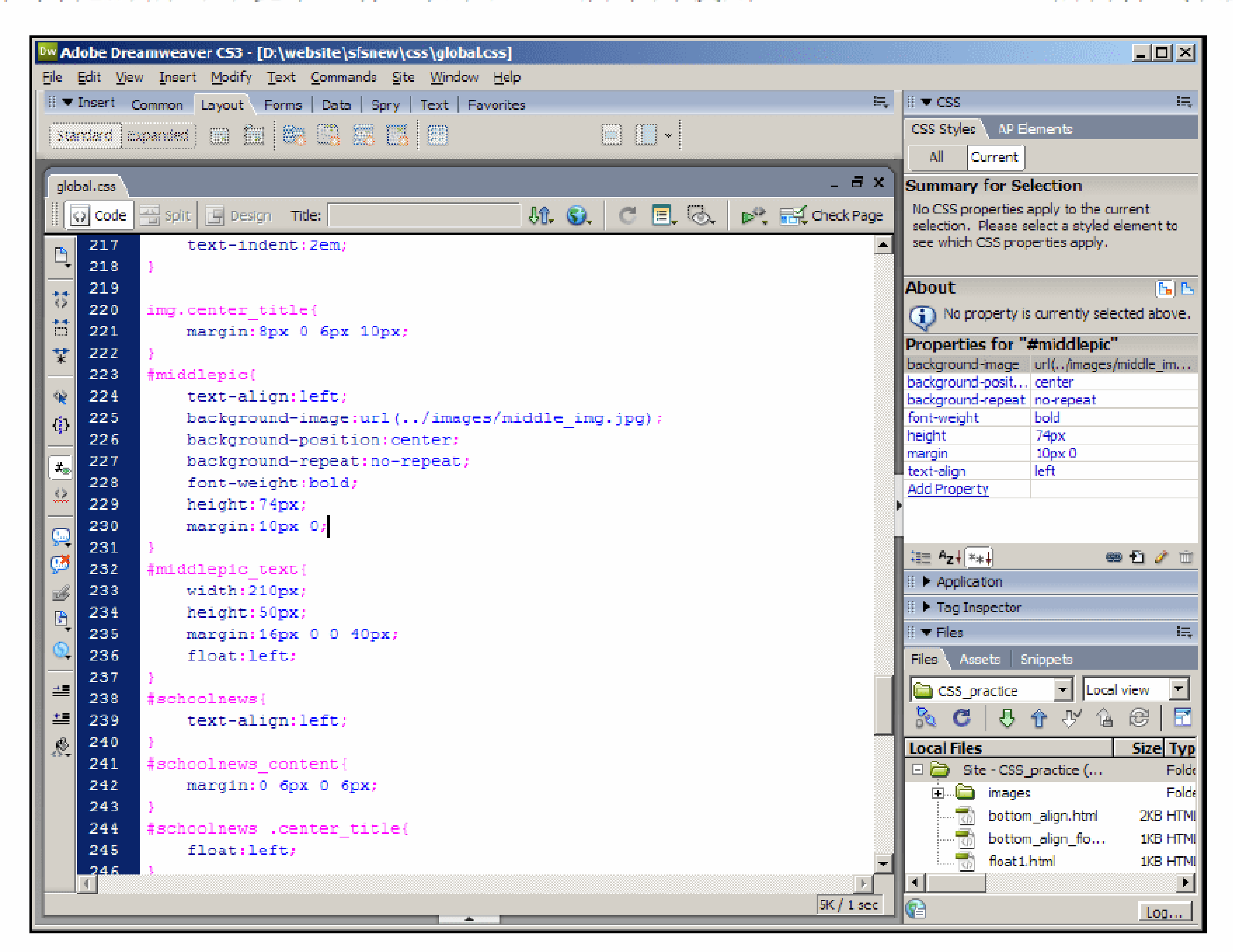
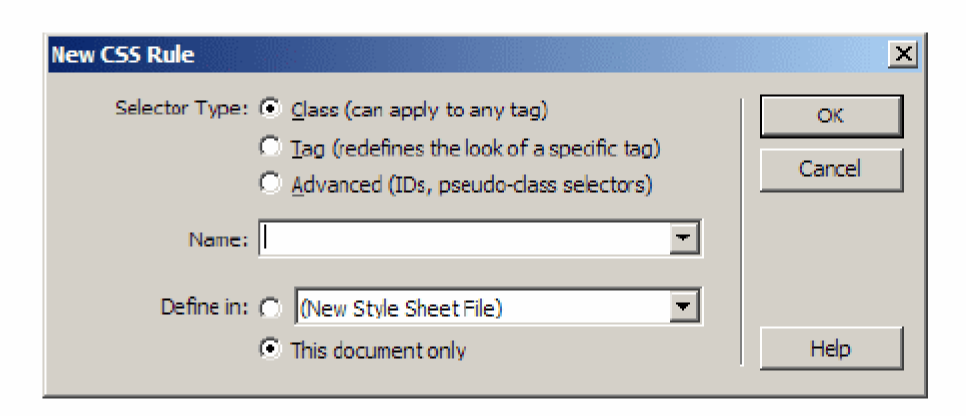


图 3-13　使用 Adobe Dreamweaver CS3 编辑样式表

除了直接编辑 CSS 代码文件以外，Dreamweaver 还提供了另一种方式来创建和编辑样式，如图 3-14 所示为该软件提供的样式创建和编辑窗口，只需要直接输入对应样式的属性值即可完成代码的编写工作。



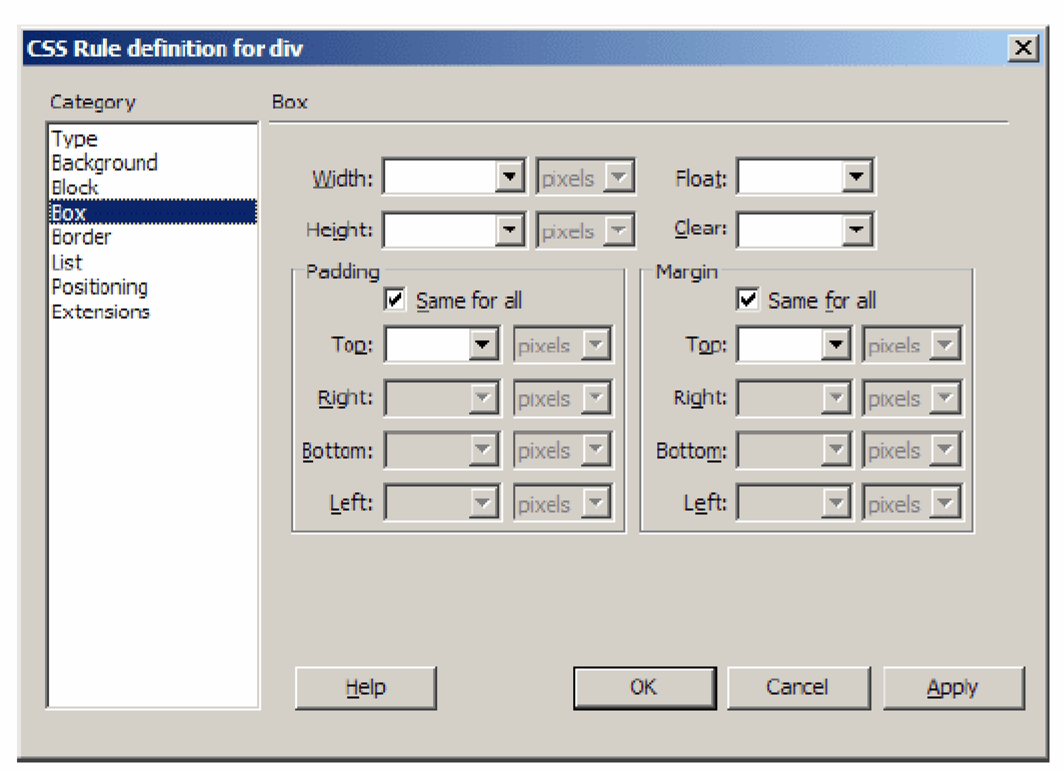


图 3-14　Dreamweaver 提供的 CSS 样式创建和编辑窗口

微软公司的推出的 Expression Web 也是一款功能强大的 Web 开发工具(见图 3-15)。是微软 Expression Studio 的产品之一。

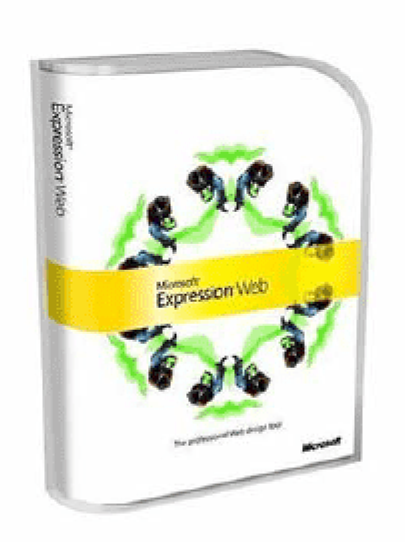




图 3-15　Microsoft Expression Web 产品

该产品取代了微软的前一代网页制作工具 FrontPage。配合 Visual Studio 开发环境，Web 设计人员和后台开发人员可顺利地进行协同工作。使用强大的接口工具可直接处理位置、缩放、边界和填补。样式应用可精确地控制 CSS 样式的产生方式，以及产生于何处，并使用样式产生器来进行完善的样式设计和有效率的样式编辑。

如图 3-16 所示为 Expression Web 的界面，通过 CSS 属性窗口可编辑属性值，样式管理窗口列出所有选择符，单击后可以预览样式效果。

如图 3-17 所示为添加新样式的界面，你可以确定选择符、样式生成的位置，编辑样式属性并能预览效果。



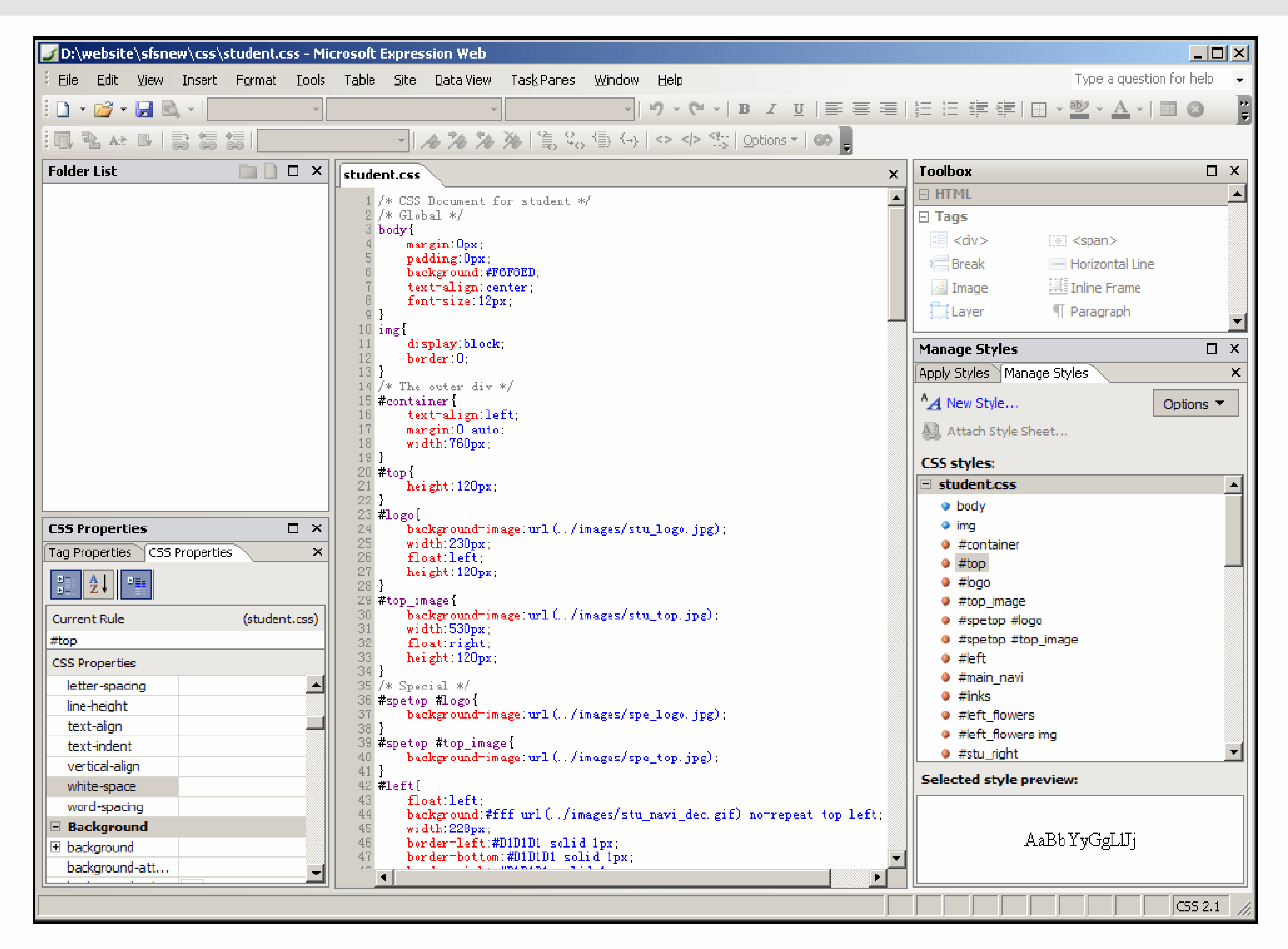


图 3-16　Expression Web 的界面

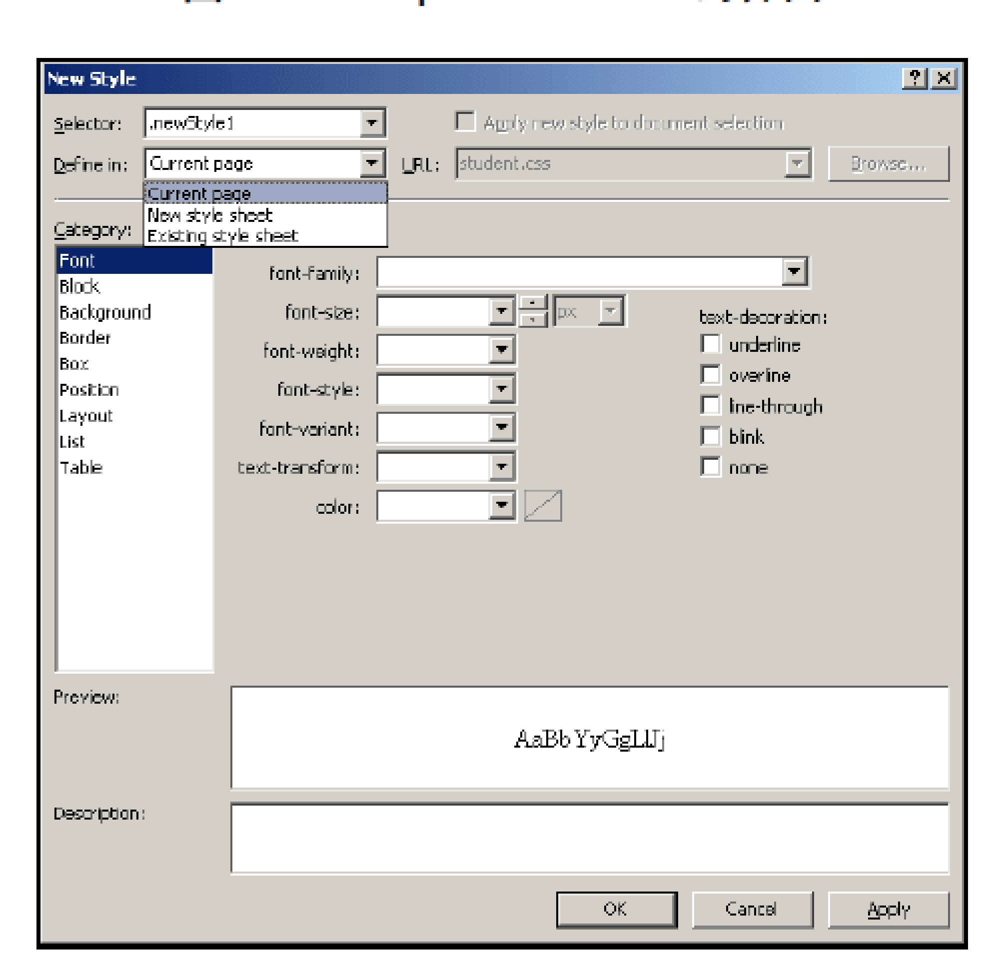
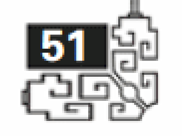


图 3-17　Expression Web 添加新样式的界面

3.9.2　辅助工具

想要一次编写出正确的样式代码几乎是不可能的，你需要不断地在不同版本的浏览器中验证效果。当页面元素过多、样式表过于复杂的时候，你很难发现错误所在。借助于一些辅助工



具，你可以全面深入分析页面结构、快速定位错误。下面介绍一些常用的辅助工具。

1. IE Developer Toolbar

Internet Explorer Developer Toolbar 是由微软公司发布的基于 IE 的开发辅助工具。它让开发人员能够深入探索和理解页面，帮助开发者更好地创建 Web 应用。安装后可以在 IE 中快速分析页面结构。该工具条可集成在 IE 窗口中，也可以浮动窗口的形式存在。

IE Developer Toolbar 的特性如下：

- 可浏览和修改 Web 页的文档对象模型。
- 通过多种技术方式定位、选定 Web 页上的特定元素。
- 查看(X)HTML 对象的类名、ID，以及类似链接路径、Tab 顺序、快捷键等细节。
- 描绘表格、单元格、图片或选定标记的轮廓。
- 即时重定义浏览器窗口的大小。
- 清空浏览器缓存和 Cookie，被清除项可从所有对象或给定域中选择。
- 直接访问关联的 W3C 规范参考、IE 开发组 Blog 或其他资源。
- 显示设计时标尺，帮助对齐对象。

如图 3-18 所示为 IE Developer Toolbar 显示页面 DOM 结构和其样式信息的情形。

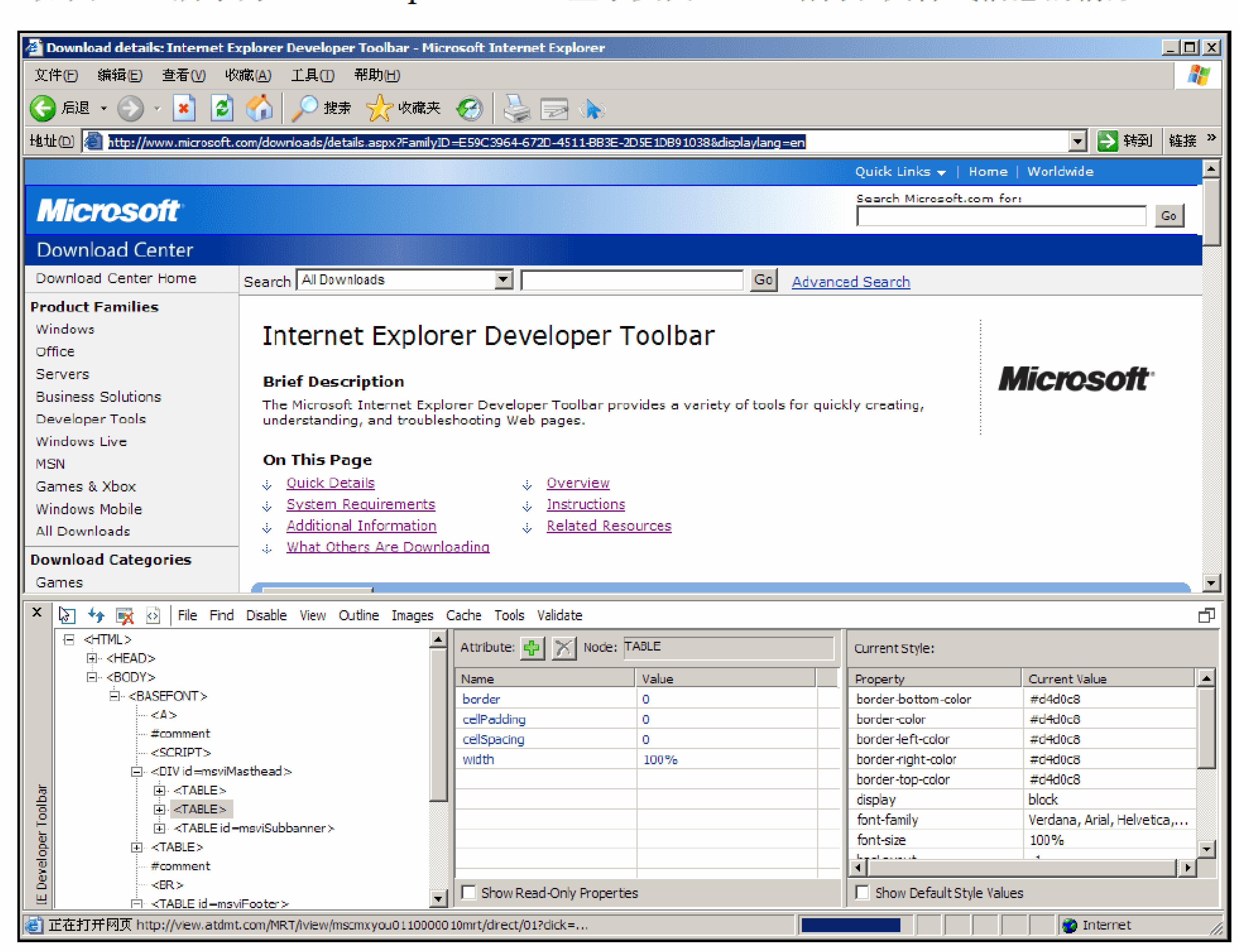
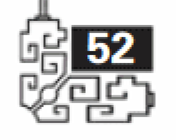


图 3-18 使用 IE Developer Toolbar 浏览 DOM 结构和样式信息

图 3-19 展示了该工具提供的标尺功能。通过它可查看页面某一元素的宽度和高度。



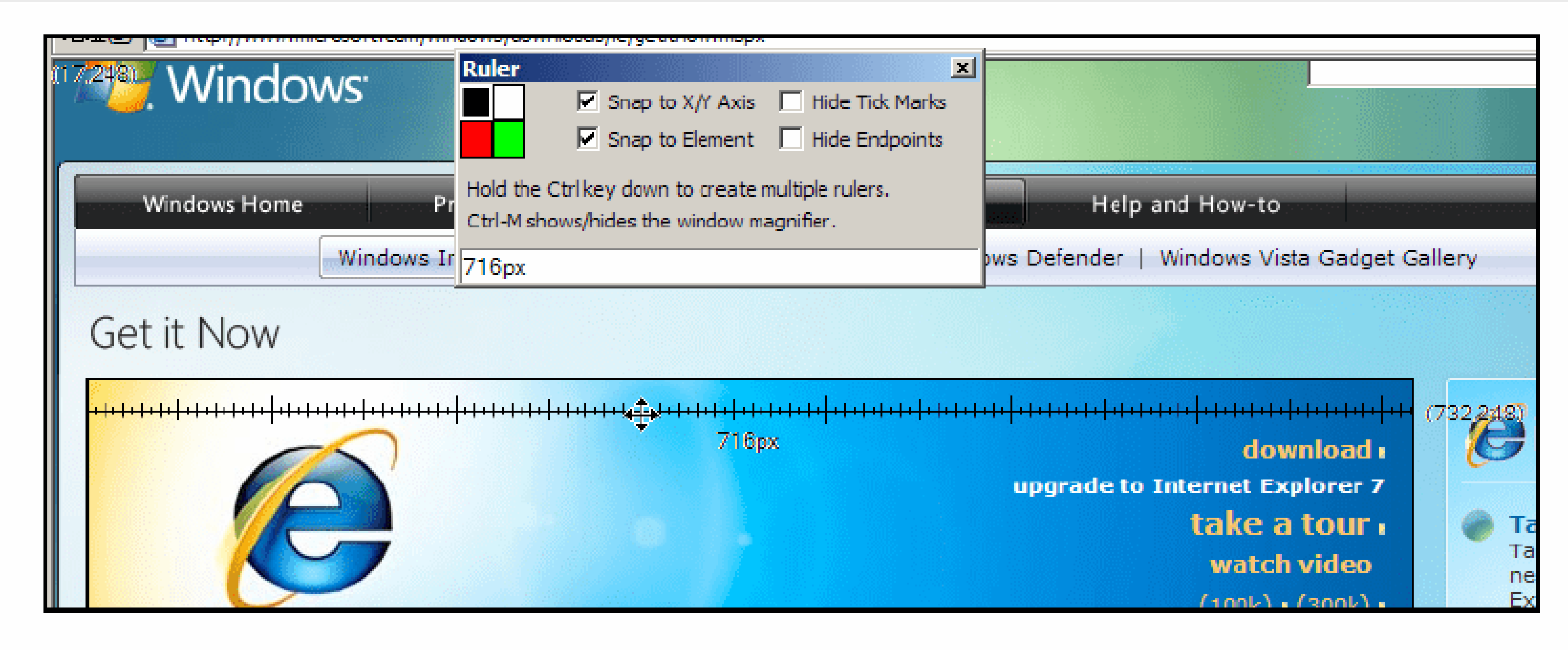


图 3-19　使用 IE Developer Toolbar 提供的标尺功能

Internet Explorer Developer Toolbar 的网址如下：

```
http://www.microsoft.com/downloads/details.aspx?familyid=E59C3964-672D-4511
-BB3E-2D5E1DB91038&displaylang=en
```

2. DOM Inspector

DOM Inspector 是集成在 Firefox 浏览器中的一个小工具。它可以用来检查和编辑页面元素。你可以浏览每一个 DOM 结点并查看其样式属性。图 3-20 为使用 DOM Inspector 工具查看页面元素样式信息的情形。

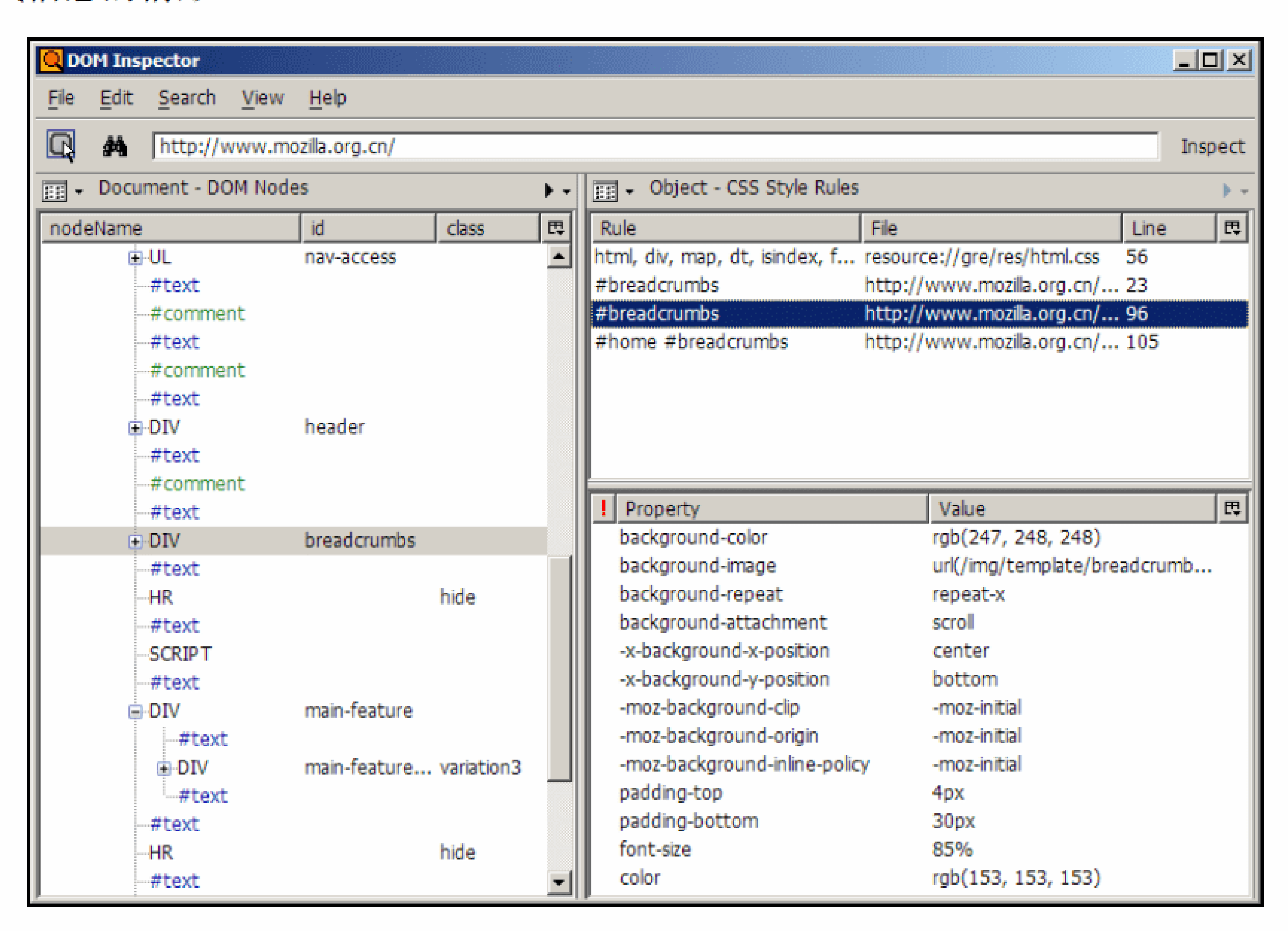


图 3-20　使用 DOM Inspector 查看 DOM 结构和样式信息

3. Firebug

Firebug(http://www.getfirebug.com/)是一款基于 Firefox 浏览器的开发辅助工具。你可以编辑、监视 CSS 样式。它还能以图形化方式显示盒模型信息。如图 3-21 所示为 Firebug 显示页面

某元素的盒模型信息。

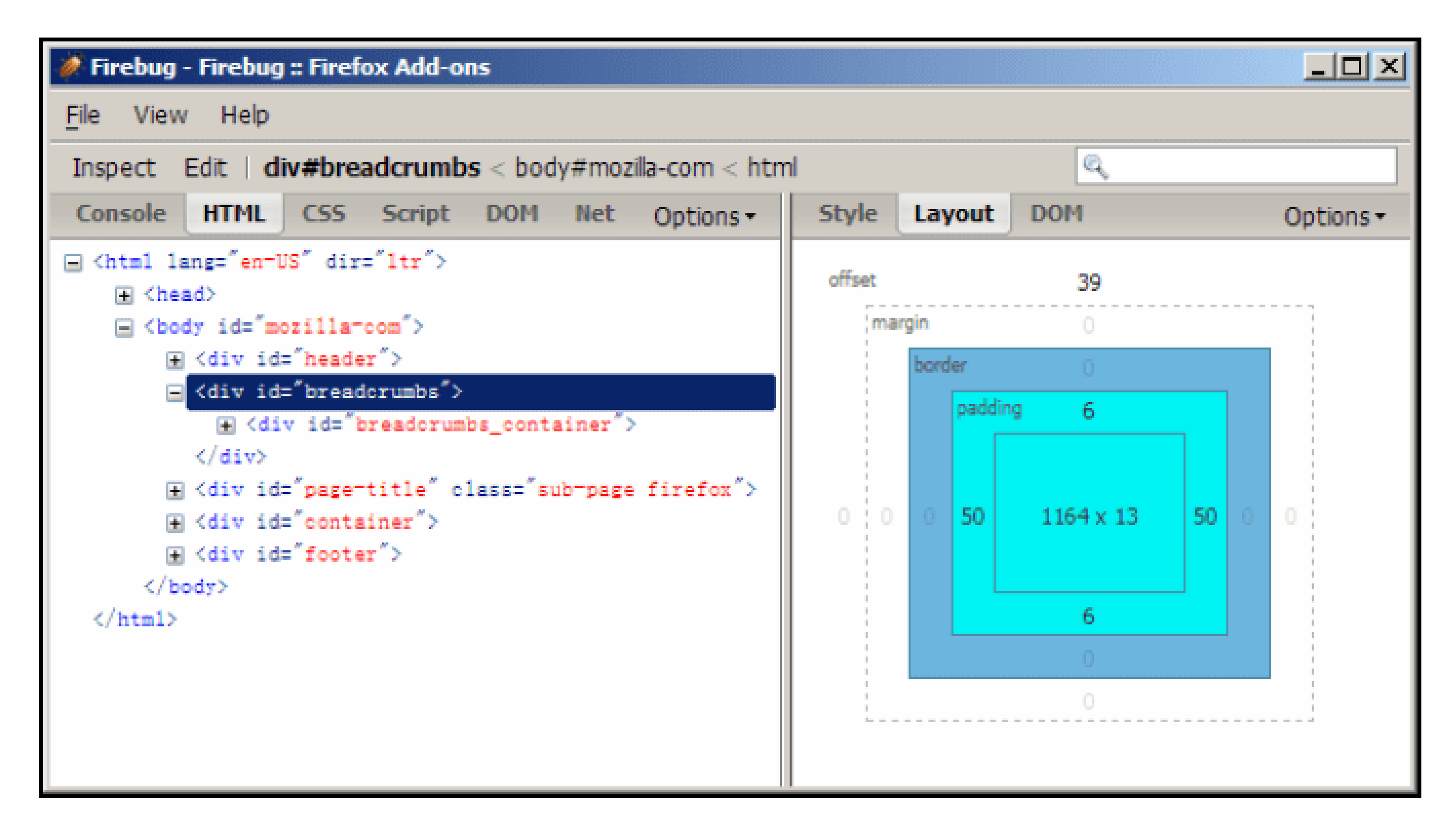


图 3-21 使用 Firebug 图形化显示元素盒模型信息

Firebug 提供了浏览、实时编辑页面元素样式信息的功能，在 Style 窗口中会显示元素的所有相关样式信息，开发者可以禁用已有属性、添加新的属性和样式声明，如图 3-22 所示。

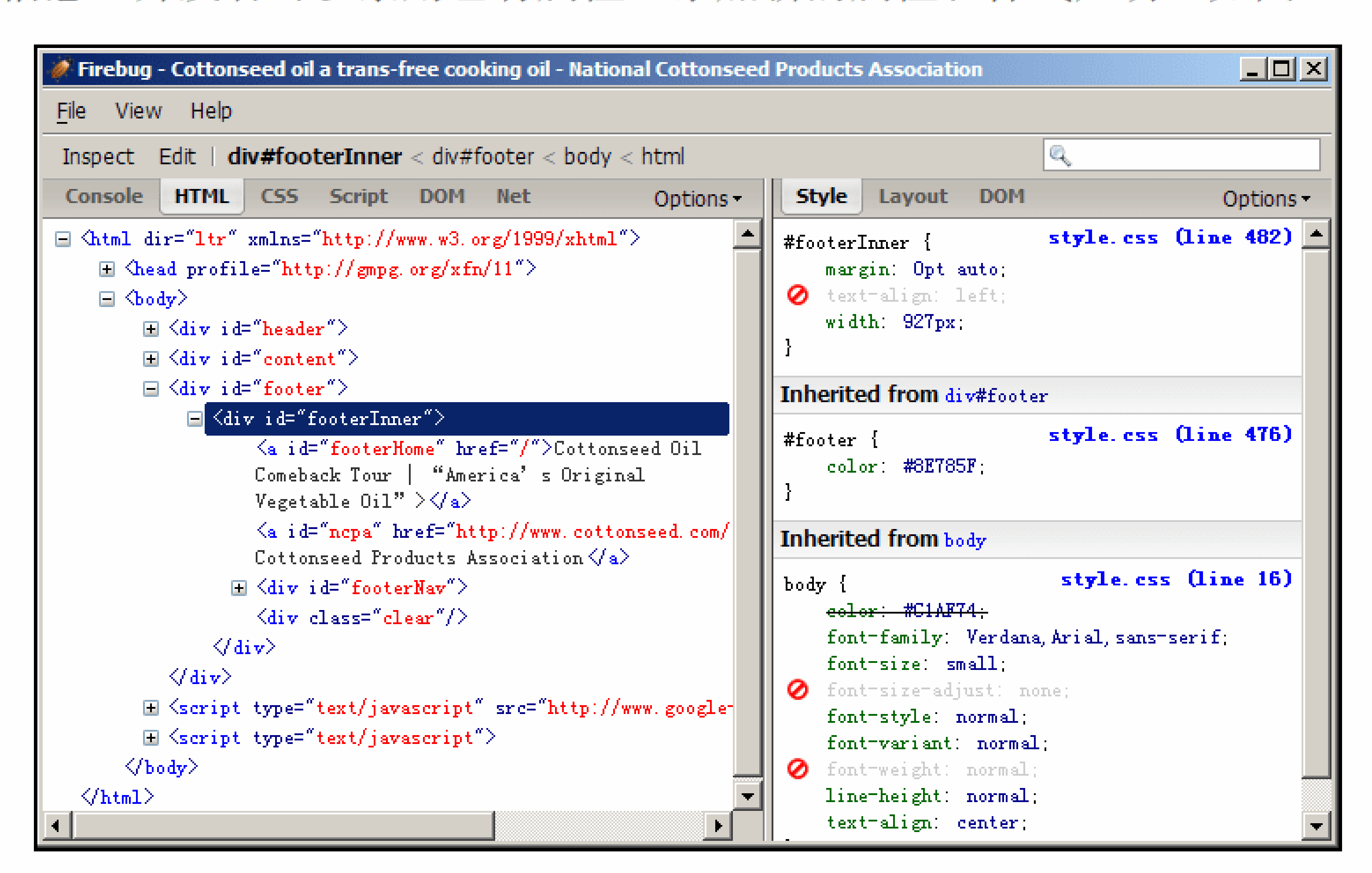


图 3-22 使用 Firebug 浏览、编辑元素样式

4. Web Developer

Web Developer 也是基于 Firefox 开发的开发辅助工具，安装后会在浏览器中添加一个新的工具条，如图 3-23 所示。

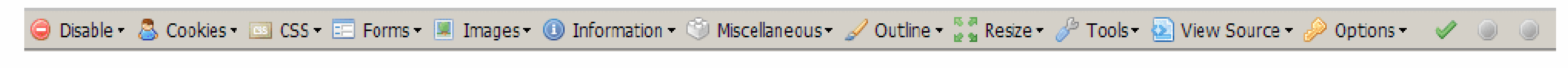


图 3-23 Web Developer 工具条

在单击页面中的某个元素后，该工具将立刻显示该元素的样式信息，并支持实时修改样式。如图 3-24 所示为当用鼠标单击某个元素后(红色方框表示)，显示出元素的 DOM 结构信息和样



式表文件内容。

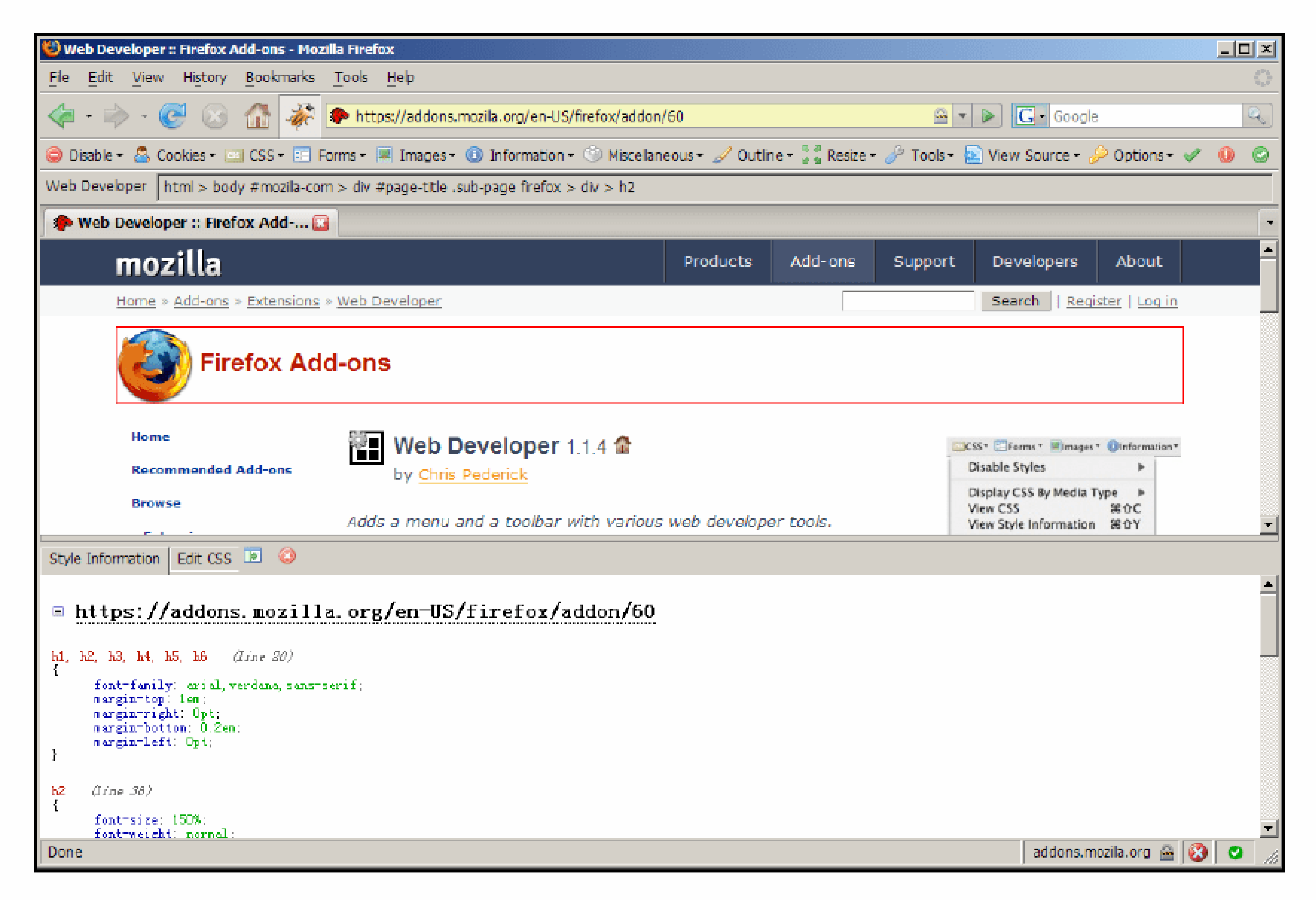


图 3-24　使用 Web Developer 浏览样式信息

如图 3-25 所示为正在使用 Web Developer 的实时修改样式的功能，在现有的样式中添加一条 float:right 规则，让 Firefox 图标移到右侧。规则添加完毕后页面即可显示修改之后的效果。

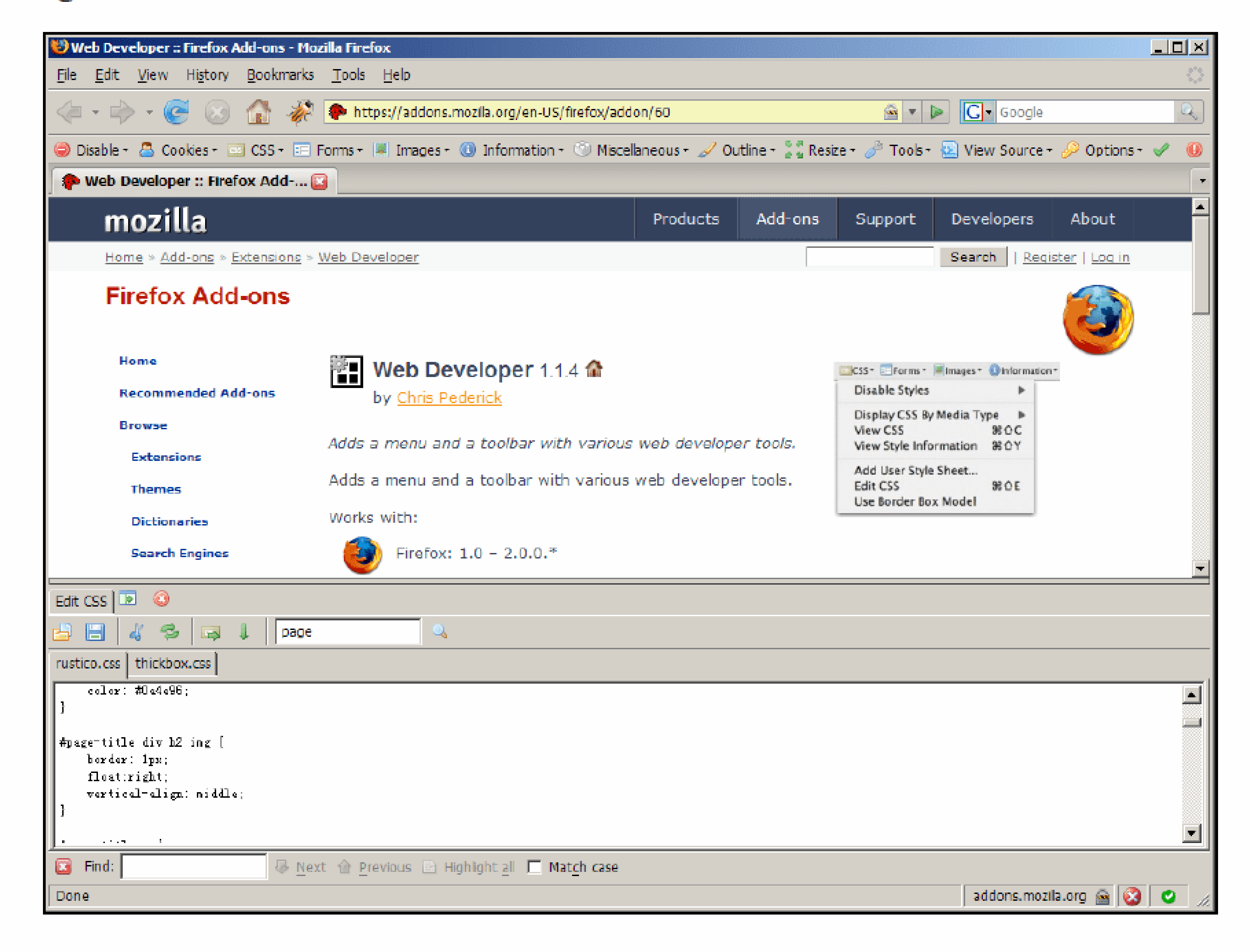


图 3-25　实时修改 CSS 代码并即时预览

Web Developer 的下载地址如下：

https://addons.mozilla.org/en-US/firefox/addon/60

3.10 小　　结

CSS，英文全称为 Cascading Style Sheet，中文译为层叠样式表，它是一种遵循特定层叠规则的用于定义文档外观样式的语言。

CSS 使得页面内容和样式信息分离成为可能。通过 CSS，Web 设计者不仅能够设计出美仑美奂的页面效果，还能减少冗余的样式信息，加快页面的下载速度，改善代码的可读性和可维护性。

CSS 规范由国际万维网联盟(W3C)定义。目前广泛使用的规范版本为 CSS 2.1，新版的 CSS3 还处于工作草案或候选建议状态。

有三种方式可以将 CSS 应用到(X)HTML 页面中：①内联方式——直接利用元素的 style 属性添加样式信息；②嵌入方式——将页面中所有样式信息集中在 style 元素中；③外部方式——通过(X)HTML 的<link>标记将样式表文档与(X)HTML 文档关联起来。

对于小型网站来说，它所包含的页面数量少且样式较为单一，可以使用一个 CSS 文件来控制所有页面的样式。而对于那些大中型网站来说，可以将 CSS 代码模块化，保存为多个样式文件。

一条 CSS 规则由选择符和声明部分组成，声明部分包含 0 个或多个声明，声明之间用分号隔开，每个声明由属性和值组成，二者用冒号分隔。选择符指明哪个元素将使用所定义的样式，声明部分指明所定义的属性和属性值。这就是组成 CSS 规则的全部内容。

虽然不像一般编程语言要求的那样——在实际开发过程中代码应有 20%的注释内容，但是在 CSS 文档中添加适当的注释信息可以增强代码的可读性，便于开发和管理。在代码中使用一致的风格同样会使代码便于阅读和维护，并能在视觉上吸引人。

使用文本编辑器就可以完成编写 CSS 代码的工作，这也是作者建议初学者使用的工具。CSS 编辑器和 Web 开发工具所附带的 CSS 编辑功能会增添许多高级特性，比如代码关键字彩色显示、样式兼容性检查、样式预览和可视化设计，它们会帮助 Web 设计者提高工作效率。

一次性编写出完美的 CSS 样式几乎是不可能的，在设计过程中很大一部分时间都会花在寻找问题根源、解决问题和如何兼容不同分辨率和不同版本的浏览器上。本章介绍的几款开发工具会提供如下高级功能：分析页面 DOM 结构、实时查看和更改元素样式、图形化显示盒模型信息，从而帮助设计者更快地定位错误、分析原因、提高工作效率。

从下一章开始，将进入本书的第二篇——CSS 核心原理，第 4 章将介绍有关 CSS 选择符的概念。

第二篇　CSS 核心原理

第 4 章　选　择　符

选择符是 CSS 的基本内容之一，它的作用在于使样式能够针对特定的页面元素。发挥选择符的作用对于充分掌握和灵活运用 CSS 至关重要。本章将介绍 CSS 中选择符的含义、使用方法以及各种浏览器的支持程度。

本章主要内容

- 什么是选择符，它有什么作用
- 选择符的模式和语法
- 类型选择符
- id 和 class 选择符，如何正确使用
- 什么是伪类和伪元素，如何使用
- 其他类型的选择符

4.1　选择符的模式和语法

选择符是 CSS 规则的组成部分，其作用是告知浏览器样式规则应该应用到页面的哪些元素中，只有理解和掌握了选择符的含义和用法，充分发挥其应有的作用，才能使用好样式表。

4.1.1　模式

选择符实际上就是一系列的模式，只要页面中有完全匹配这个模式的元素存在，样式就会应用到该元素上。比如选择符 p 就是一种最简单的模式，它是(X)HTML 中的元素名，匹配页面中所有的 p 元素。

4.1.2　语法

简单的选择符可能只需要一个元素名称，而复杂的选择符可能会涉及许多内容，我们现在就要介绍选择符的语法。

CSS 规则的语法并不复杂，CSS 2.1 规范是这样定义的：

- 一个简单选择符包含一个通用选择符或是一个类型选择符，它的后面跟有 0 个或多个 id 选择符、属性选择符或者伪类，三者的顺序可以任意。

- 一个选择符是由一个或多个简单选择符组成的，中间用连接符相隔。
- 连接符可以是空白、大于号(>)或者加号(+)，空白可以出现在连接符及其周围的简单选择符之间。

刚看完上述语法说明时，你会感觉有点不知所云，因为里面出现了许多新概念。没有关系，本章余下的内容将会详细介绍这些新概念，阅读完本章全部内容后你就能掌握并使用选择符。

4.2　基本选择符

CSS 规则的语法要求每个选择符必须包含一个通用选择符或者一个类型选择符，我们把这两类选择符称为基本选择符，它们是构成选择符的基础部分。

4.2.1　通用选择符

通用选择符用星号(*)表示，它选择页面中所有的元素，比如：

```
* {
    background-color:white;
}
```

上面这个规则将所有元素的背景颜色设置为白色。在一个简单选择符中，如果通用选择符不是唯一的组成部分，那么我们就可以省略掉它。比如以下两种写法就是相同的：

```
*.title{
    font-size:16px;
    color:red;
}

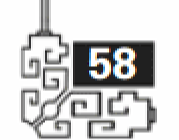
.title{
    font-size:16px;
    color:red;
}
```

上面的第一条规则中，通用选择符后面跟了一个属性选择符(我们后面会介绍它使用了 class 属性)，在第二条规则中，星号被省略掉了。

4.2.2　类型选择符

类型选择符，又叫元素选择符或标记选择符，是所有选择符中最常见的。选择符的全部内容就是某一种(X)HTML 元素名称，它匹配文档中对应的元素。例如：

```
h1 {
    color:red;
}
```



上面的规则应用到所有的<h1>标记，并指定其颜色为红色。类型选择符简单易用，但也有很大的限制。若要进行更具体、更复杂的选择，就要用到后面将介绍的选择符了。

4.3 组 选 择

假如多个元素具有相同的 CSS 样式，为了减少代码冗余，就可以使用组选择，考虑如下 CSS 代码：

```
h1{
    font-family:Tahoma, Arial, "宋体";
    font-size:14px;
}
h2{
    font-family:Tahoma, Arial, "宋体";
    font-size:14px;
}
h3{
    font-family:Tahoma, Arial, "宋体";
    font-size:14px;
}
p#title{
    font-family:Tahoma, Arial, "宋体";
    font-size:14px;
}
```

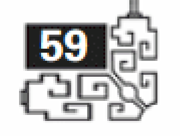
h1、h2、h3 和 id 为 title 的 p 元素的样式是一样的，但这样写会使 CSS 代码变得很臃肿，我们可以将选择符组合在一起写，例如上面的规则可以写成：

```
h1, h2, h3, p#title{
    font-family:Tahoma, Arial, "宋体";
    font-size:14px;
}
```

这 4 个选择符被组合成为一个选择符，代码量大大降低了。使用组合选择符可以很方便地将含有相同样式的元素组合在一起。

4.4 id 和 class 选择符

id 和 class 选择符可以说最常用的两种选择符了，前者根据赋给元素的 id 值进行匹配，而后者根据元素的 class 属性进行匹配。合理使用 id 和 class 选择符会大大增强样式代码的利用率，减少冗余。



4.4.1 id 选择符

每个(X)HTML 元素都包含一个 id 属性，该属性值是唯一的，可以唯一标识一个元素。通过 id，我们就可以选择更具体的元素。id 选择符由井号“#”加 id 值组成，考虑如下(X)HTML 代码：

```
<h1 id="bulletinTitle">公告</h1>
<p id="bulletinContent">很高兴我的博客和大家见面了</p>
<h1 id="articleTitle">如何同时使用 IE6 和 IE7? </h1>
<p id="articleContent">Web 开发人员通常需要在多种类型的浏览器上测试页面效果，但是微软的 IE6 和 IE7 不能并存，页面设计者不得不使用多台电脑或安装多系统或虚拟机来解决问题。下面我就介绍如何在同一个系统下同时使用两个版本的 IE 浏览器。</p>
```

我们给每一个 h1 元素和 p 元素分配不同的 id，通过 id 选择符，就可以更精确地控制样式所应用到的元素，比如只要求前两个元素应用样式：

```
h1#bulletinTitle{
    color:white;
    background-color:silver;
}
p#bulletinContent{
    color:yellow;
    background-color:maroon;
}
```

效果如图 4-1 所示，只有特定元素使用了样式，其他元素不受任何影响。

公告

很高兴我的博客和大家见面了

如何同时使用IE6和IE7?

Web开发人员通常需要在多种类型的浏览器上测试页面效果，但是微软的IE6和IE7不能并存，页面设计者不得不使用多台电脑或安装多系统或虚拟机来解决问题。下面我就介绍如何在同一个系统下同时使用两个版本的IE浏览器。

图 4-1 通过 id 给特定元素添加样式

由于 id 具有唯一性，因此可以将#号前的元素名称省略掉(实际上，#号前添加了一个省略了星号的通用选择符)，但增加元素名称可增加代码的可读性。

注意： 包含元素类型的 id 选择符与不包含元素类型的 id 选择符并不是完全等同的，它们拥有不同的确定度。有关确定度的概念会在第 6 章中进行介绍。



4.4.2　class 选择符

class 选择符是属性选择符的一种，由于它的使用频率较高且通常和 id 选择符一起使用，我们把它放在这里进行介绍。

(X)HTML 元素包含 class 属性，与 id 属性的不同之处在于，class 属性值不要求具有唯一性，即多个元素可以拥有相同的 class 值，因此 class 选择符可以同时选择多个元素。class 选择符由英文句点“.”加类名组成(稍后我们会看到这种写法是类属性选择符的一种特殊形式)。

将上一小节的代码稍做修改如下：

```
<h1 id="bulletinTitle" class="title">公告</h1>
<p id="bulletinContent" class="content">很高兴我的博客和大家见面了</p>
<h1 id="articleTitle" class="title">如何同时使用 IE6 和 IE7? </h1>
<p id="articleContent" class="content">Web 开发人员通常需要在多种类型的浏览器上测试
页面效果，但是微软的 IE6 和 IE7 不能并存，页面设计者不得不使用多台电脑或安装多系统或虚拟机来
解决问题。下面我就介绍如何在同一个系统下同时使用两个版本的 IE 浏览器。</p>
<h1 class="title">Internet Explorer 7 新特性</h1>
<h2 class="titleLevel2">选项卡式浏览</h2>
<p>选项卡式浏览一向是 Internet Explorer 的弱项，只有通过外挂程序解决，但效果不是很理想。
因此，选项卡式是 Internet Explorer 用户最迫切需要的功能。利用选项卡式浏览，可以在一个浏
览器窗口中管理多个页面，极大地提高了浏览效率，避免了开启多个窗口而使桌面变得杂乱不堪。使用
Internet Explorer 7，只需单击选项卡即可在网站间切换。
</p>
```

CSS 代码如下：

```
h1.title{
    color:white;
    background-color:silver;
}
p.content{
    color:yellow;
    background-color:maroon;
}
h2.titleLevel2{
    color:gray;
}
```

从图 4-2 可以看出，class 属性为 title 的两个 h1 元素的文字为白色，背景为银灰色，class 属性为 content 的两个 p 元素的文字为黄色，背景是暗红色。但样式不影响最后一个 p 元素，因为它没有指定 class 属性。最后，样式代码把 class 值为 titleLevel2 的 h2 元素中的文字设为灰色。在上述的例子里面，我们给同一类型的元素指定了相同的类名，当然也可以给不同类型的元素指定相同的类名。请看下面的(X)HTML 代码：

```
<h1 id="newfeature" class="important">Internet Explorer 7 新特性</h1>
<h2 id="zoom" class="important">页面缩放</h2>
<p>在原来版本的 IE 浏览器中，用户可以对文字进行缩放操作，新版 Internet Explorer 7 在这一
基础上增强了缩放功能。该功能不仅可以缩放文字，而且能对<span class="important">页面中的
```

```
任意图形</span>进行缩放。 网站上较小的文本或缩略图图像现在都可以放大浏览了。用户只需按住
Ctrl 键同时滚动鼠标滚轮即可实现缩放功能。
</p>
```

公告

很高兴我的博客和大家见面了

如何同时使用IE6和IE7?

Web开发人员通常需要在多种类型的浏览器上测试页面效果，但是微软的IE6和IE7不能并存，页面设计者不得不使用多台电脑或安装多系统或虚拟机来解决问题。下面我就介绍如何在同一个系统下同时使用两个版本的IE浏览器。

Internet Explorer 7 新特性

选项卡式浏览

选项卡式浏览一向是Internet Explorer的弱项，只有通过外挂程序解决，但效果不是很理想。因此，选项卡式是 Internet Explorer 用户最迫切需要的功能。利用选项卡式浏览，可以在一个浏览器窗口中管理多个页面，极大地提高了浏览效率，避免了开启多个窗口而使桌面变得杂乱不堪。使用Internet Explorer 7，只需单击选项卡即可在网站间切换。

图 4-2　使用 class 选择符给多个元素指定样式

CSS 代码如下：

```
*.important{
    font-weight:bold;
    color:red;
}
```

上述规则使用了通用选择符加类选择符，它匹配所有 class 属性为 important 的元素，这里星号可以省略。在图 4-3 中，相应元素中文字显示为红色粗体。

Internet Explorer 7 新特性

页面缩放

在原来版本的IE浏览器中，用户可以对文字进行缩放操作，新版Internet Explorer 7 在这一基础上增强了缩放功能。该功能不仅可以缩放文字，而且能对页面中的任意图形进行缩放。 网站上较小的文本或缩略图图像现在都可以放大浏览了。用户只需按住Ctrl键同时滚动鼠标滚轮即可实现缩放功能。

图 4-3　使用 class 选择符给不同类型元素指定样式

4.4.3　多重 class

由上节的内容可知，class 属性和 id 属性的不同之处在于，id 属性是唯一的，而同一 class 属性可赋给多个元素。除此之外，一个(X)HTML 元素还可以属于多个 class。这就是多重 class(multiple class)。下面的 a 元素就使用了多重 class，它同时属于 class1 和 class2：

```
<a class="class1 class2" href="/">链接</a>
```



使用两个 class 选择符就可以分别指定它们的样式：

```
a.class1{
    font-size:12px;
}
a.class2{
    color:red;
}
```

上面两条规则使用了 class 选择符，但它们还没有充分发挥 class 选择符的功能，因为它们不仅指定了 a 元素的样式，其他任何属于 class1 或 class2 的 a 元素均会应用这个样式，这样选择符不能精确匹配 a 元素。而使用多重选择符(Multiple Selector)就能达到这个目的，它可以精确匹配那些同时属于多个 class 的元素，比如下面这两条规则只会把上面的 a 元素文字设为红色，其他任何只属于 class1 或 class2 的元素都不会受到影响：

```
a.class1.class2{
    color:red;
}
```

或：

```
a.class2.class1{
    color:red;
}
```

使用多重 class 和多重选择符的优势在于，你可以设计一个基本 class 的样式作为默认值，若要添加新样式，则可以设计一个新 class 样式并追加进来。原来设计的基本样式不需要任何改动，从而形成一种层次化的设计。请看下面的示例。

(X)HTML 代码：

```
<div class="box">
    <div class="title">产品列表</div>
    <div class="content">
        <h3>Web Professional</h3>
        <p>Web Professional 是一款可视化的 Web 开发软件，它能快速地创建优美的 Web
        页面，提供了强大的网站管理功能。</p>
    </div>
    <div class="content">
        <h3>Web Express</h3>
        <p>Web Express 是一款轻巧的可视化 Web 开发工具，使用它可快速创建您所需要的
        Web 页面。</p>
    </div>
</div>
```

CSS 代码如下：

```
body{
    font-family:Tahoma, Arial, "宋体";
    font-size:small;
}
```

```
div.box {
    border:1px solid #E2DFD0;
    margin-bottom:1em;
}
div.box div.title {
    padding:0.3em;
    padding-left:0.6em;
    background: #F2F0DD;
    font-size:0.9em;
    font-weight:bold;
}
div.box div.content{
    margin:0.3em;
}
div.box div.content *{
    margin:0;
}
div.box h3{
    font-weight:bold;
    font-size:0.9em;
    padding:0.5em;
    color:#0033CC;
}
div.box p{
    font-size:0.9em;
    padding:0.5em;
}
div.content + div.content{
    border-top:1px dotted #E2DFD0;
}
```

如图 4-4 所示为样式的效果。



图 4-4　在运用多重选择符之前的一般情况下的样式

这些规则涉及的样式在这里不做具体讲解，因为这不是我们的重点。假如某一段时间该 Web 页面需要重点突出产品列表部分(比如更换一个更为醒目的背景色)，以后会再恢复正常。为了达到此目的，你当然可以直接修改原有的 CSS 代码，恢复时再修改回来。但是多重选择符会减少你一定的工作量，你可以事先使用多重选择符添加突出显示的样式：



```
/* css for featured box */
div.box.featured {
    background-color: #FFFDBA;
}
div.box.featured div.title {
    background-color: #FFF94E;
}
```

当需要突出显示这部分内容时，你只需要添加一个新的 class 属性 featured 到 div 中：

```
<div class="box featured">
```

图 4-5 中的背景色已经变为醒目的黄色。

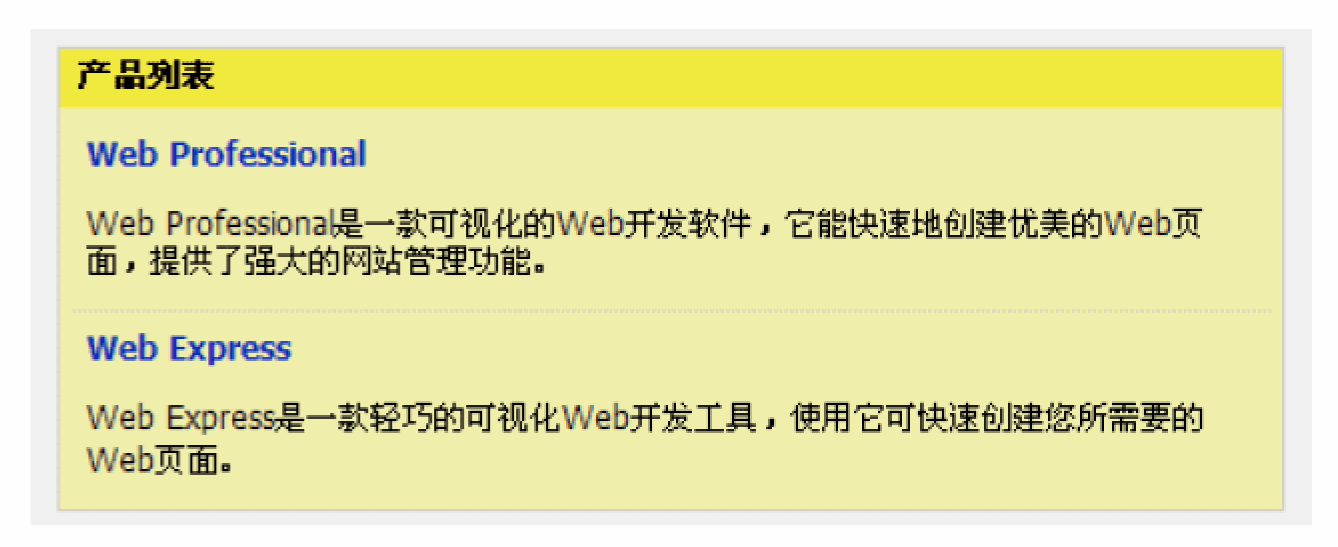


图 4-5　追加新的 class 属性并使用多重选择符

若要恢复默认的样式，只需要去掉 class 属性相应的部分即可。假如某一段时间要重点突出某个单一的产品，同样你可以先写好样式的代码，需要时只要追加新的 class 属性到相应的元素即可。这种方式比直接修改 CSS 代码方便了许多，也避免了由于修改的遗漏而可能导致的错误。

注意：Internet Explorer 6 及以前版本对多重 class 支持效果并不理想。上述示例中的选择符 div.box.featured 应该只能匹配 class 属性为“box featured”的 div 元素，但是在 IE 中它还会匹配 class 属性为“featured”的 div 元素。因此，在命名多重 class 属性时要倍加小心，否则在 IE6 中会出现意想不到的效果。

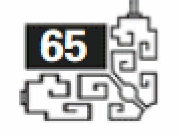
4.4.4　用 id 还是 class

你可以给任意数量、任意类型的元素指定 class 属性。而一个 id 属性值只能赋值给唯一的元素。若你指定一个 h1 元素的 id 值为 first-title，那么任何其他元素都不能再指定这个 id 值。

当需要给页面某个单独的元素(比如登录部分的标题文字或显示天气预报的侧栏)指定样式时，你就该使用 id 选择符。当需要给一组元素指定样式时，class 选择符则是你的首选。

4.4.5　class 和 id 属性的命名规则

(X)HTML 元素的 class 和 id 属性值可用来标识某些或某个特定的元素。开发人员在命名这



些属性值时应尽量保持名称简单明了且符合一定的命名规范。虽然命名规范的约定与开发人员的个人习惯有关，但是如果命名方式太随意或者太个性化，就会导致 CSS 样式代码难以阅读和维护。下面就介绍一些比较常见的 class 或 id 属性值的命名方式。

1. 根据样式含义命名 class 属性

含有 class 选择符的 CSS 规则往往将样式应用在若干个(X)HTML 元素上，这些规则通常会包含一些共有的样式，这时可以根据它们的具体含义来命名，比如下面这些 class 属性值。

- cRed：小写 c 表明颜色(color)，后面加具体颜色名称。
- fBold：小写 f 表示字体(font)，后面表示粗体显示。
- noBorder：表明元素不使用边框。
- left：表明元素向左浮动。

2. 根据元素所在页面位置来命名 class 和 id 属性

使用描述元素所在页面位置的词汇来对其进行命名也是一种比较常见的命名方式。有时也和元素的用途结合在一起使用。这种方式使得 class 以及 id 的名称如下面所示。

- header：表示页面顶端区域。
- footer：页面底脚部分区域。
- leftSide：页面左侧区域。
- rightCol：页面右侧栏目区域。

这种根据元素位置来对属性进行命名的方式简单有效，但也有它的弊端。比如页面放置新闻的区域在开始设计时放在了左侧，你可以起个类似于 leftSide 的 id 名。但是有一天页面改版，需要把新闻区域移动到右侧，leftSide 这个名称就显得极为不合适了，继续使用可能造成理解的混乱，这时你就不得不重新命名。

为了避免这种混乱的发生，可以采用下面介绍的命名方式。

3. 根据使用样式的元素用途来命名 class 和 id 属性

某些(X)HTML 片段在页面中不止出现一次，它们会被大量地重复使用，比如表示更多信息的链接文字(通常为“更多”二字)、固定宽度的广告条区域等。这时可以按照下面的方式根据(X)HTML 元素的用途来命名 class 属性。

- ad468：表示宽度为 468px 的广告条区域。
- area：表示一个区域。
- more：表示显示更多信息的链接区域。
- newsHeadline：表示所有新闻的标题区域。
- inputText：表示所有需要用户输入的文字。
- codeTextArea：表示显示代码的文字区域。

对于那些只出现一次的(X)HTML 元素，适合添加 id 属性，可以按照以下方法对 id 属性进行命名。

- mainNav：表示一级菜单导航区域。
- subNav：表示二级菜单导航区域。
- copyright：表示显示版权信息的区域。



这种命名方式意味着元素的位置信息与其内容是完全分离的。这些名字同根据位置命名的方式一样简单易懂，但它们描述了页面元素的用途而非位置。这使得(X)HTML 代码更加符合使用纯粹的结构化标记的初衷，即开发人员可以在不改变标记的情况下对不同的显示格式进行处理。开发人员还可以对元素的用途进行如下的细化，从而实现一种结构化的命名方式。

- login：表示登录区域。
- loginTitle：表示登录区域的标题。
- loginTxt：表示登录区域的文字。
- loginInput：表示登录区域的输入区域。

这些名称都是在 login 作为基础名称之上扩展而来，由外层逐步深入细化到内层。开发人员很容易在原有的基础上对样式进行修改。

命名方式不仅仅只有以上提到的三种，但它们是最常见的。在实际开发过程中通常把三种命名方式结合在一起使用。在表达页面基本结构的时候可以使用诸如 header、footer 等表示位置的名称，在细节部分使用根据用途命名的名称，对于那些需要多次使用的样式，则可以使用样式的含义进行命名。

提示：程序代码通常会遵循 Pascal 或 camel 命名法，在命名元素的 class 和 id 属性时也可以采用这两种命名法的一种。Pascal 命名法指名称中每个单词的首字母大写，如 MainNav、LoginInput；camel 命名法指名称中第一个单词的第一个字母不是大写，后面的单词首字母需要大写，如 subNav、sideBar。除此之外，单词之间也可以由连字符“-”或下划线“_”相连，单词首字母全部小写，比如：search-input，header-logo。

4.5　伪元素和伪类

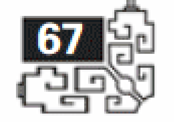
除了类型选择符、ID 选择符和 class 选择符外，CSS 还允许使用伪元素和伪类选择符。

伪元素是一种在(X)HTML 标记中并不实际存在的抽象元素。比如，标记语言并没有一种机制可以访问某个元素的首个字符或首行内容。而 CSS 伪元素则创建了这样的虚拟元素，通过这种虚拟元素就可以访问并控制其样式。

4.5.1　:first-line 和:first-letter 伪元素

使用:first-line 伪元素可以添加只应用于某元素首行的样式。该元素必须是块级(Block)、内联块(Inline-block)、表格标题(Table Caption)或表格单元(Table Cell)，所添加的样式属性也是有限的：

- 字体属性
- 颜色(color)
- 背景属性
- 词间距(word-spacing)



- 字间距(letter-spacing)
- 文字装饰(text-decoration)
- 垂直对齐(vertical-align)
- 文字变形(text-transform)
- 行距(line-height)

下面这条规则将 p 的首行文字的显示比例增大了：

```
p:first-line: {font-size:130%;}
```

:first-letter，顾名思义，就是指元素的首个字符，虽然在(X)HTML 代码中并没有对应的标记，但你可以在 CSS 中使用它。这个伪元素使得我们很容易定义、更改首字符的样式，而不用定义新的标记。同:first-line 一样，应用:first-letter 元素需是块级元素、内联元素、列表项、表格标题或表格中的单元格。除了可以使用:first-line 规定的属性外，:first-letter 还可以使用如下属性(集)：

- 浮动(float)
- 边距(margin)
- 填充(padding)
- 边框(border)

下面这个示例运用了:first-line 和:first-letter 伪元素。

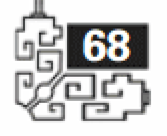
(X)HTML 代码：

```
<h1 class="title">IE 1.0 的诞生</h1>
<p class="textbody">1995 年 8 月，微软家族中一个日后呼风唤雨的成员呱呱落地，它就是
Microsoft Ineternet Explorer 1.0 了，但当时微软并未对这个新生儿给予多大的照顾，因为比
尔盖茨正沉浸在 1995 年 1 月发布的 Windows 95 所带来的巨大喜悦里，这才是他们的重头戏。</p>
<p class="textbody">这个其貌不扬功能简陋的 IE 1.0 只是基于 NCSA Mosaic 的简单"修改版"。
没有增加太多的新特性，不支持 Java，不支持插件，浏览速度也很缓慢。Netscape 浏览器还没有把这
个小家伙看在眼里，据说当年给 IE1.0 做研发工作的只有几个人而已。</p>
<p class="textbody">这时 IE1.0 还是一个单薄的程序，并未引起太大的反响。人们仍然用 Mosaic
或者 Netscape 来浏览网页。不过面对 Netscape 如日中天的气势，微软似乎也嗅到了一丝危机。</p>
```

样式表部分如下：

```
p.textbody:first-letter{
    font-size:xx-large;
    background-color:black;
    color:red;
    font-weight:bold;
}
p.textbody:first-line{
    background-color:silver;
}
```

结果如图 4-6 所示(IE7 浏览器)。



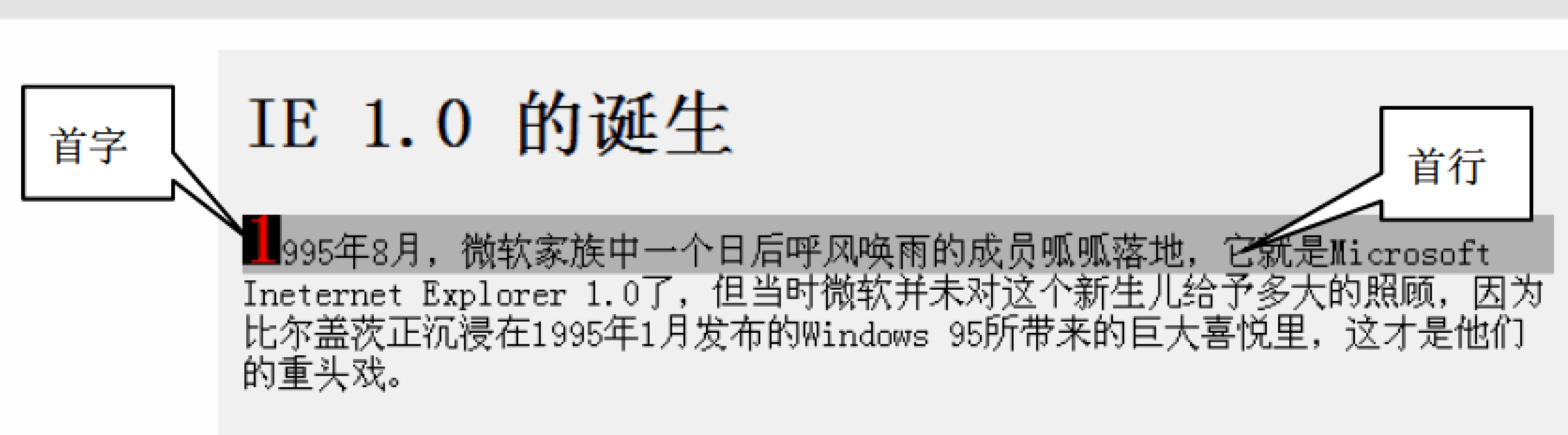


图 4-6　使用:first-letter 和:first-line 伪元素设计首字和首行样式(IE7 浏览器)

注意： IE6 及以前版本不支持:first-line 和:first-letter 伪元素，因此上述代码在其中不会产生任何效果。另外，图 4-6 所示结果取自 IE7，它和 Firefox 的显示结果也存在细微的差别。若要兼容 IE6，就需要手动在首字周围添加一个标记，比如<span>。再添加一个 class 属性，比如 class="first_letter"，然后在规则里面添加一个 class 选择符。例如：span.first_letter，最后把样式填写到声明当中。对于:first-line 伪元素，由于该伪元素的具体范围在显示之前是不确定的，事先不可能添加一个标记，因此目前也没有很好的解决办法。

4.5.2　:before 和:after 伪元素

:before 和:after 伪元素可以用来确定要产生的内容放置在哪里，可以在元素的开头或结束位置。请看下面的示例：

```
p:before{
    content:"重要通知！";
}
p:after{
    content:"不要迟到。";
}

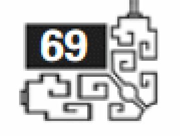
<p>今日下午 2 点将在报告厅举行专场学术报告会，请准时参加。</p>
```

结果如图 4-7 所示，p 元素内容前后分别添加了“重要通知！”和“不要迟到。”几个字。

重要通知！今日下午2点将在报告厅举行专场学术报告会，请准时参加。不要迟到。

图 4-7　使用:before 和:after 伪元素添加新内容(Firefox 浏览器)

:before 和:after 伪元素不可以在一条规则中同时使用，但可以分别放在两条不同的规则中，这样既能在元素之前也能在其后添加新的元素。另外，它还可以同其他类型选择符一起相结合使用。



注意： 目前为止，任何版本的 IE 浏览器均不支持:before 和:after 伪元素，图 4-7 显示的结果取自 Firefox 浏览器。若要兼容 IE 浏览器，需要手动添加新元素并设定相应的样式。

4.5.3　:first-child 伪类

:first-child 伪类匹配满足下面条件的元素：该元素是其父元素的首个子元素。在下面这个示例中，选择符匹配那些是其父元素第一个孩子的 p 元素。

```
p:first-child{
    text-indent:2em;
}
```

上述规则会使得下面(X)HTML 代码中的 p 元素首行有两个字符的缩进：

```
<div>
    <p>本段文字是父元素 div 的首个子元素，:first-child 伪类会匹配本元素，因而会有两
    个字符的缩进。</p>
</div>
```

但是下面代码中的 p 元素就不会应用此样式了，因为它不是 div 的首个子元素。

```
<div>
    <h1>我才是首个子元素</h1>
    <p>本段文字不是父元素 div 的首个子元素，:first-child 伪类不会匹配本元素，因而不
    会应用任何样式。</p>
</div>
```

效果如图 4-8 所示。

　　本段文字是父元素div的首个子元素，first-child伪类会匹配本元素，因而会有两个字符的缩进。

我才是首个子元素

本段文字不是父元素div的首个子元素，first-child伪类不会匹配本元素，因而不会应用任何样式。

图 4-8　:first-child 伪类的作用

4.5.4　有关链接的伪类

CSS 提供:link 和:visited 伪类来区分一个链接是尚未访问还是已经访问过，这两种状态相互排斥，一个链接不可能同时处于两种状态，只能是二者之一。

有关链接的使用我们将在第 10 章中做详细的介绍。

4.5.5　有关用户动态行为的伪类

网页是用来与用户交互的，用户的某些行为会在页面元素中表现出来，比如将鼠标悬停在某个元素上或是在某个元素上按下鼠标左键。CSS 提供了三种伪类来处理，有关动态行为伪类，也放到第 10 章中进行具体的讲解。

4.5.6　:lang 伪类

(X)HTML 元素有个 lang 属性，它指定了该元素所使用的语言。语言规定了文本方向、可使用的字体、自动阅读器的发音规则等。浏览器通过该属性可以正确显示出与语言相关的符号。

CSS 提供:lang 伪类来匹配使用某种语言的元素。比如：

```
:lang(de){
    color:red;
    font-weight:bold;
}
```

以上规则的作用是，将所有德文内容的文字设为红色粗体，我们只需要在(X)HTML 元素中添加 lang 属性，并设置其属性名为“de”即可，比如：

```
<p>德国人早晨见面打招呼会说：<span lang="de">Guten tag</span>。</p>
```

效果如图 4-9 所示(该图取自 Firefox 浏览器，IE 不支持)。

德国人早晨见面打招呼会说：**Guten tag**。

图 4-9　使用:lang 伪类(Firefox 浏览器)

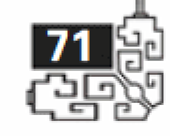
4.6　与元素关系相关的选择符

4.6.1　后代选择符

有时，网页设计者需要为某个元素的某种类型的后代元素指定样式，而不想影响该元素之外的同类型元素，比如把某个 div 中的 p 元素的行距设为标准行距的 1.5 倍。使用后代选择符就可以做到，后代选择符由两个或多个选择符组成，中间用空格分隔。选择符“A B”将匹配任何是 A 的后代的 B 元素。

下面的(X)HTML 代码中有两个 div 元素，每个有唯一的 ID 标识且各包含一个 p 元素。p 分别是两个 div 的子元素，当然也是后代元素。

```
<div id="weatherinfo">
    <p>今日天气：晴；最高气温：25 摄氏度；风力：2-3 级。</p>
</div>
```



```
<div id="datetime">
    <p>今天是 2008 年 8 月 8 日。</p>
</div>
```

下面的规则将天气部分文字设为橙色，背景为蓝色；日期部分的文字设为白色，背景设为黑色：

```
div#weatherinfo p{
    color: orange;
    background-color: blue;
}
div#datetime p{
    color: white;
    background-color: black;
}
```

效果如图 4-10 所示。

今日天气：晴；最高气温：25摄氏度；风力：2-3级。

今天是2008年8月8日。

图 4-10　使用后代选择符匹配特定元素

需要说明的是，后代元素不仅包含直接子元素，也包含任何下一层次的元素，即所有的后代。所以不管元素嵌套次数有多少，后代选择符都会匹配。

4.6.2　子选择符

与后代选择符类似，子选择符只匹配那些作为某个特定元素的子元素，但不包括子元素的子元素。子选择符由两个或更多的选择符组成，中间用符号“>”分隔。选择符“A > B”表示匹配那些是 A 的子元素的 B 元素。

下面这个例子将 ID 为 notice 的 div 元素里所有 p 子元素行距设为普通行距的 1.5 倍：

```
div#notice > p{
    line-height:1.5em;
}
```

子选择符也可以与后代选择符组合使用，例如：

```
div#middle div > p a {
    color:red;
    font-weight:bold;
}
```

其中，a 是 p 的后代元素，p 是 div 的子元素，而该 div 又是 id 为 middle 的 div 的后代元素。上述规则会匹配下面的 a 元素：



```
<div id="middle">
    <div id="navigation">
        <p>欢迎访问小晖子工作室网站，点击<a href="http://www.huistd.com">此处</a>
        进行浏览。
        </p>
    </div>
</div>
```

最终效果是使链接部分的文字为粗体红字。

4.6.3　邻接兄弟选择符

邻接兄弟选择符的形式如下：E1 + E2，其中 E2 是要选择的元素。匹配该选择符的元素需要满足以下条件，E1 和 E2 元素有共同的父元素且 E1 是 E2 的直接前驱，即 E1 是紧挨在 E2 之前的那个元素。此时非元素结点将会被忽略掉(比如文本结点或注释信息)。下面这个示例使得紧挨在 h1 元素之后的 p 元素顶部有 10 个像素的边距：

```
h1 + p{
    margin-top:10px;
}
```

下面我们通过一个更为具体的例子来体会邻接兄弟选择符的作用。考虑如下(X)HTML 代码：

```
<dl>
    <dt>层叠样式表 (CSS)</dt>
    <dd>英文全称为 Cascading Style Sheet，由 W3C 创建的一种语言，用于描述结构化文
    档的表达效果。</dd>
    <dd>更多内容请参考：
        <a href="http://www.w3.org/Style/CSS/">
            http://www.w3.org/Style/CSS/
        </a>
    </dd>
    <dt>规则 (rule)</dt>
    <dd>CSS 中指定样式的指令，由选择符和一段声明组成。</dd>
    <dt>选择符 (selector)</dt>
    <dd>CSS 规则的一部分，是一种模式，匹配应用样式的标记元素。</dd>
    <dt>声明 (declaration)</dt>
    <dd>CSS 规则的一部分，声明可以是空也可以包含一种或多种样式。声明包含属性名和属性值。
    </dd>
    <dt>属性 (property)</dt>
    <dd>在 (X) HTML 元素上设置的一个量，对该元素进行样式设计。属性具有名称和值。</dd>
</dl>
```

下面的样式表出现了一些新属性：

```
body{
    font-family:Arial, "宋体";
    font-size:small;
}
dt {
    font-weight: bold;
    font-size:1.2em;
}
dd + dt{
    border-top: 1px solid black;
    padding-top:1em;
}
dt + dd{
    font-size:0.9em;
    padding-top:0.6em;
}
```

不用担心，这里先介绍它们的含义，后面的章节中会有更为详细的介绍。

上述规则首先定义 body 元素所使用的字体族为 Arial 和宋体，字号为 small，接着设定 dt 文字为粗体，大小为 1.2 个单位。接下来的选择符选择紧挨着 dd 元素后的 dt 元素，设置上边框为 1 个像素宽度、solid 样式且为黑色，接下来的属性是 dt 元素上方填充的高度，这里设置为 1 个单位大小。这条规则在页面上显示的效果就是每条术语之间有一条水平线和一定的空隙。最后一条规则选择紧挨在 dt 后的 dd 元素，设定字体大小为 0.9 个单位，上方填充 0.6 个单位。如图 4-11 所示为 IE7 显示的结果。



图 4-11　使用邻接兄弟选择符选择特定元素(IE7 浏览器)



注意：以上提到的子选择符和邻接兄弟选择符在 IE6 中均不识别。为了获得最大兼容性，应尽量使用后代选择符，如果还得不到所需的效果，只好手动添加 class 属性并使用 class 选择符。

4.7　属性选择符

本章最后一节我们介绍属性选择符，它可以说是 CSS 选择符里使用最为灵活的一种。通过它我们就可以选择那些具有某种属性的所有元素，将样式应用到其中。

4.7.1　属性选择符的匹配方式

CSS 允许我们针对元素的某个属性进行选择，属性选择符的匹配方式有以下 4 种。

- [att]：匹配任何使用 att 属性的元素，不管其属性取值如何。
- [att=val]：匹配 att 属性值为 val 的元素。
- [att~=val]：匹配满足以下条件的元素：元素的 att 属性的值需是一个由空格分隔的单词列表，其中一个单词必须是 val。若要使用该选择符，val 值不能包含空格，否则会被视为多个单词。
- [att|=val]：匹配满足以下条件的元素：元素的 att 属性的值需是一个由连字符“-”分隔的单词列表，val 是其中的第一个单词。

注意：*属性选择符使用起来灵活度很高，但遗憾的是 IE6 及以前的版本不支持属性选择符。为了获得最大兼容性，只能给元素附加 class 属性，在 CSS 中附加 class 选择符并填写相应的样式。*

4.7.2　属性选择符示例

表 4-1 列举了一些属性选择符和一些(X)HTML 元素，并指出是否匹配。

表 4-1　属性选择符匹配测试

CSS 选择符	(X)HTML 片段	是否匹配
*[lang]	<p lang=“fr”>	是
p[title]	<div title=“click me”>	否
table[rules=“cols”]	<table rules=“cols”>	是
table[rules~=“cols”]	<table rules=“cols”>	是
table[rules|=“cols”]	<table rules=“cols”>	是
span[class=“bar”]	<span class=“foo bar”>	是
span[class~=“bar”]	<span class=“foo bar”>	是
span[class|=“bar”]	<span class=“foo bar”>	否
[lang=“zh”]	<p lang=“zh”>	是
[lang~=“zh”]	<p lang=“zh”>	是
[lang|=“zh”]	<p lang=“zh”>	是

续表

CSS 选择符	(X)HTML 片段	是否匹配
[lang="zh"]	<p lang="zh-cn">	否
[lang~="zh"]	<p lang="zh-cn">	否
[lang|="zh"]	<p lang="zh-cn">	是

下面看一个使用属性选择符的具体例子。假设你需要在页面里添加一些文件下载的链接，你想通过使用不同图片来标识不同类型的文件。(X)HTML 的 a 元素表示一个链接，它有个 type 属性，该属性指明链接所指内容的类型，我们可以赋给该属性表明文件类型的属性值，再使用属性选择符匹配相应的链接元素。

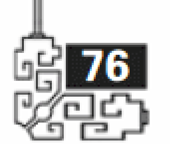
事先要准备好 4 张表示不同文档类型的图片，命名为 typedoc.gif、typepdf.gif、typexls.gif 和 typetxt.gif，它们分别表示 Word 文档、PDF 文档、Excel 文档和文本文档。

(X)HTML 代码如下：

```
<h1>勘误表下载，请选择适合你的文档类型。</h1>
<a href="errata.doc" type="doc">下载</a>(Word 文档格式)<br />
<a href="errata.pdf" type="pdf">下载</a>(PDF 格式文档)<br />
<a href="errata.xls" type="xls">下载</a>(Excel 格式文档)<br />
<a href="errata.txt" type="txt">下载</a>(文本文档)
```

CSS 代码如下：

```
body{
    font-size:small;
    font-family:"Times New Roman", "宋体";
}
h1{
    font-size:140%;
}
a{
    padding-left:40px;
    padding-top:10px;
    padding-bottom:10px;
    height:40px;
    font-size:small;
    line-height:3em;
    background-position:left;
    background-repeat:no-repeat;
}
a[type="doc"]{
    background-image:url(typedoc.gif);
}
a[type="pdf"]{
    background-image:url(typepdf.gif);
}
a[type="xls"]{
    background-image:url(typexls.gif);
```



```
}
a[type="txt"]{
    background-image:url(typetxt.gif);
}
```

最终结果如图 4-12 所示。

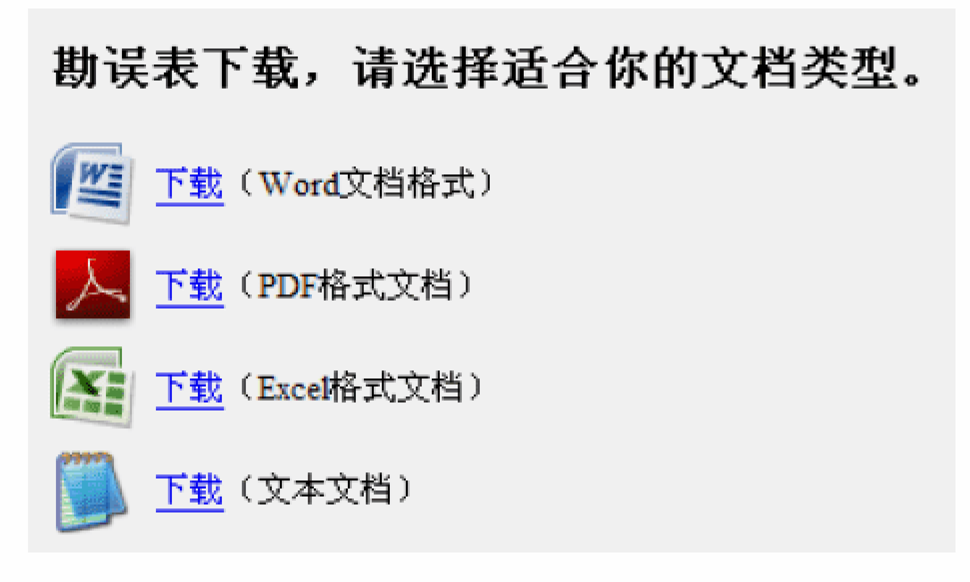


图 4-12　通过属性选择符设置特定的样式(Firefox 浏览器)

延伸： 下面的网址总结了目前各种流行的浏览器对不同版本的 CSS 选择符的支持程度：http://www.quirksmode.org/css/contents.html

4.8　小　　结

本章介绍了 CSS 规则中比较重要的部分：选择符。它指明了页面中哪些元素应用样式。

一个简单选择符包含一个类型选择符或一个通用选择符，后面跟有 0 个或多个 id 选择符、属性选择符或伪类，三者的顺序可以任意。一个选择符由一个或多个简单选择符组成，中间用连接符相隔。连接符可以是空白、大于号(>)或者加号(+)，空白可以出现在连接符和其周围的简单选择符之间。

通用选择符可以在任何元素上设置样式；类型选择符用来匹配(X)HTML 中的特定元素；class 和 id 选择符根据元素的 class 属性和 id 属性来选择。合理使用和命名 class 和 id 属性会提高开发效率，减少维护的工作量。

伪元素和伪类可以选择一些实际并不存在的元素。后代选择符、子选择符和邻接兄弟选择符会根据元素的层次结构进行匹配。属性选择符匹配那些满足指定要求的元素。

在下一章中，我们将会介绍有关 CSS 度量的内容。

第 5 章　CSS 中的度量

每一条 CSS 的声明都包含了属性和属性值，从背景颜色到字体大小，从元素高度到段落行距，丰富多彩的样式都是依靠赋予属性不同的值来实现的。不同的属性可能需要不同类型的属性值，比如长度定义用整数，字体定义用字符，有关颜色的定义用颜色关键字或 RGB 颜色值。有些属性值还需要添加适当的单位。

本章将介绍 CSS 属性值所使用的各种值类型及其所带单位的含义和使用方法。

本章主要内容

- CSS 中度量的概念
- CSS 中的值
- CSS 中值的单位
- 相对单位和绝对单位

5.1　值 的 类 型

在定义样式表时，不同属性所需的属性值的类型也可能会不同，下面对各种值逐一介绍。

5.1.1　整数和实数

整数和实数都是以十进制形式表示的。例如：

```
div#adv{
    z-index:999;
}
```

数值前均可以添加“+”和“-”符号，表示正负。一些属性要求整数或实数值必需限定在一定范围内，比如非负数。

5.1.2　长度

长度是水平或垂直方向的度量，由数值类型加一个单位(如 px、em)组成，当长度为零时，单位可以省略不写。例如：

```
p.normal{
    font-size:12px;
    width:300px;
}
```

有些属性允许使用负长度，但这会增加样式设计的复杂度，也会影响兼容性。若属性不支

持负长度，则该条声明会被忽略。有关单位的用法在下一节讲解。

5.1.3　百分数

百分数由数值加百分号“%”组成。百分数通常用来表示另一个数值的相对值，这个基值可能位于同一元素的其他属性中，也可能位于某个祖先元素中。例如：

```
p#copyright{
    font-size:12px;
}
p#copyright{
    line-height:150%;       /* font-size 属性值的 1.5 倍，即 18px */
}
```

以上规则首先确定了字体大小为 12px，然后确定行距为 150%，这个百分数就是相对于同一个元素的 font-size 属性的。那么该行距大小应为 12 × 150% = 18px。

5.1.4　URL 或 URI

由“url(”加任意数量的空白，加可选的单引号(‘)或双引号(“)，后跟 URI，再加上对应的单引号(’)或双引号(”)及“)”构成。URI 即可为绝对地址也可为相对地址，例如：

```
div#top{
    background-image:url(http://www.huistd.com/images/logo.jpg);
}
```

或者：

```
div#top{
    background-image:url(images/logo.jpg);
}
```

5.1.5　颜色

颜色值可以是 CSS 规范中定义的表示颜色的关键字，或者由 RGB 值表示。

1. 关键字表示

CSS 2.1 规范提供了 17 种表示颜色的关键字(如表 5-1 所示)。

比如你希望一段文本的颜色为灰色，背景色为黑色，则需编写如下代码：

```
p{
    color:gray;
    background-color:black;
}
```

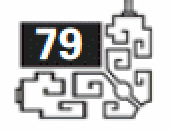


表 5-1　17 种颜色的关键字和对应的 RGB 值

关 键 字	RGB 值	说　明
aqua	#00FFFF	水绿色，浅绿色(浅蓝绿色至淡绿蓝色)
black	#000000	黑色
blue	#0000FF	蓝色
fuchsia	#FF00FF	深紫红色
gray	#808080	灰色
green	#008000	绿色
lime	#00FF00	浅绿色
maroon	#800000	栗色(暗红棕色到暗紫红色)
navy	#000080	藏青色
olive	#808000	橄榄色(一种低到中等尖亮度和低至中等饱和度的黄绿色)
orange	#FFA500	橙色
purple	#800080	紫色
red	#FF0000	红色
silver	#C0C0C0	银白色
teal	#008080	凫蓝，深青色(适度或深色的带蓝色的绿色到呈绿色的蓝色)
white	#FFFFFF	白色
yellow	#FFFF00	黄色

目前 IE、Firefox、Opera 等主流浏览器还支持 X11 颜色。它来自于 X Window 系统，它提供了 100 多个颜色名称(如 lightpink、lightseagreen 等)，这些名称已经作为 CSS3 规范的一部分。详细的颜色名称和所对应的颜色值列表可参考 CSS3 颜色模块的规范：

```
http://www.w3.org/TR/css3-color/
```

2. RGB 表示

RGB 表示红(Red)、绿(Green)、蓝(Blue)三种色光原色。三色光相互叠加以实现混色，原色的数值越高，色彩越明亮。每个原色在 0~255 之间进行取值，R、G、B 都为 0 时是黑色，都为 255 时是白色。如图 5-1 所示为 RGB 色彩模型。

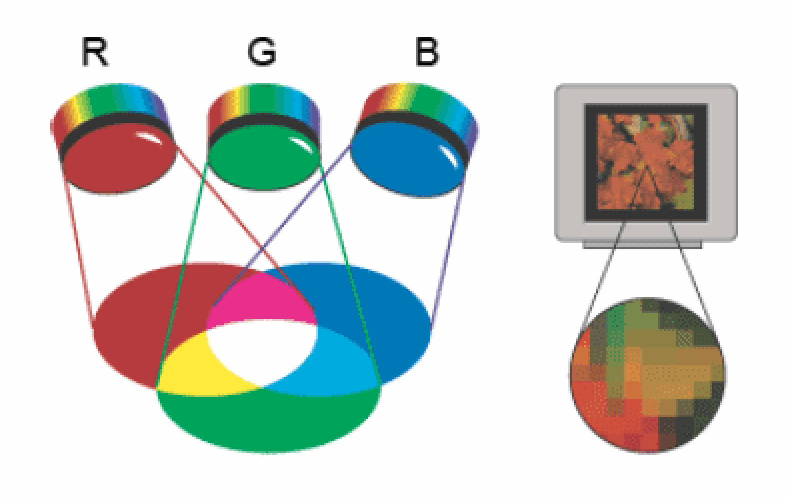


图 5-1　RGB 色彩模型(取自 Adobe 网站)

RGB 是计算机中最直接的色彩表示方法。显示器的 24 位真彩图像，就是采用 RGB 模型来精确记录色彩的。因此，CSS 也提供了这种方式来设置颜色，具体方式有以下 4 种：

(1) #RRGGBB：每种颜色值用十六进制表示，数值从 00 到 FF(字母大小写均可)。数值前要加井号“#”。例如：

```
p{color:#000000;}       /* 黑色 */
p{color:#FFFFFF;}       /* 白色 */
p{color:#55A3EB;}       /* 一种蓝色 */
```

(2) #RGB：为#RRGGBB 的简写形式，例如#CCCCCC 可简写为#CCC，#DD4455 可简写为#D45。

```
p{color:#000;}          /* 相当于#000000，黑色 */
p{color:#453;}          /* 相当于#445533，一种暗绿色 */
```

(3) RGB(十进制数值, 十进制数值, 十进制数值)：每种颜色值用十进制表示，数值从 0 到 255。例如：

```
p{RGB(255, 45, 6);}     /* 一种红色 */
p{RGB(20, 200, 120);}   /* 一种蓝绿色 */
```

(4) RGB(百分数, 百分数, 百分数)：每种颜色值用百分数表示，数值从 0%到 100%，例如：

```
p{RGB(50%, 50%, 50%);}  /* 灰色 */
p{RGB(30%, 4%, 99%);}   /* 一种蓝色 */
```

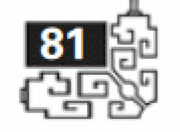
5.1.6　字符串

字符串既可用双引号也可用单引号表示，字符串中的引号需要用转义字符“\”。字符串不仅会出现在声明中，还会出现在选择符中，例如：

```
h1#hint:before{
    content: "提示: ";
}
p{
    font-family: "宋体", "Courier New";
}
a[title="a special title"]{
    text-decoration:none;
}
```

5.2　单　　位

上一节介绍了 CSS 中值的类型，其中表示长度的值需要有单位(长度为 0 时单位可以省略不写)。长度单位分为两种：表示绝对长度的单位和表示相对长度的单位。绝对长度单位一般



在输出媒介的物理大小已知的情况下才使用，而使用相对长度单位的长度值是根据另一个长度值定义的。

5.2.1 表示绝对长度的单位

当输出媒介的物理属性已知时，绝对单位才会有用处，绝对单位包括：

- in(inch，英寸，1 英寸等于 2.54 厘米)
- cm(centimeter，厘米)
- mm(millimeter，毫米)
- pt(point，点，CSS 2.1 规范中定义的点相当于 1/72 英寸)
- pc(pica，12 点字，相当于 12 点)

如果知道显示器屏幕到底有多大，或者你的页面仅仅通过打印输出，你就可以使用绝对单位，但并不是所有计算机都知道屏幕大小，所以绝对单位在实际 CSS 开发中极少使用到，我们还是看看相对长度的单位吧。

5.2.2 表示相对长度的单位

由相对长度单位定义的数值是由某个基值计算而得。相对长度单位包括 em、ex 和 px。

单位 em 表示元素自身 font-size 属性值的相对值，但是，当元素 font-size 属性值也采用 em 作为单位时，这个值就是其父元素 font-size 属性的相对值了。

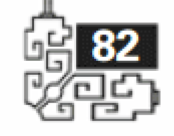
下面举例说明：

```
p#first {
    font-size:12px;
    line-height:1.2em;
}
p#second {
    font-size:12px;
    line-height:2.5em;
}

<p id="first">TopStyle 是一款 CSS 编辑软件，可以帮助我们编写符合包括 CSS2 标准在内的样式表。TopStyle 具有 CSS 定义选择功能，让你可以选取特定的浏览器或 CSS 阶层、内建的样式表检查器、内部的预览能力、以颜色标示的编辑器，以及样式预览。</p>
<p id="second">TopStyle 是一款 CSS 编辑软件，可以帮助我们编写符合包括 CSS2 标准在内的样式表。TopStyle 具有 CSS 定义选择功能，让你可以选取特定的浏览器或 CSS 阶层、内建的样式表检查器、内部的预览能力、以颜色标示的编辑器，以及样式预览。</p>
```

规则规定字体大小为 12px，两个 p 元素的行距分别为字体大小的 1.2 倍和 2.5 倍(行距是指一行的顶端与下一行顶端的距离)。

显示效果如图 5-2 所示。



TopStyle是一款CSS编辑软件，可以帮助我们编写符合包括CSS2标准在内的样式表。TopStyle具有CSS定义选择功能，让你可以选取特定的浏览器或CSS阶层、内建的样式表检查器、内部的预览能力、以颜色标示的编辑器，以及样式预览。

TopStyle是一款CSS编辑软件，可以帮助我们编写符合包括CSS2标准在内的

样式表。TopStyle具有CSS定义选择功能，让你可以选取特定的浏览器或

CSS阶层、内建的样式表检查器、内部的预览能力、以颜色标示的编辑器，

以及样式预览。

图 5-2　行距的大小是字体大小的相对值

再看一个例子。

```
body{
    font-size:small;
}
p#fstline{
    font-size:1em;
}
p#secline{
    font-size:1.5em;
}
p#thrdline{
    font-size:3em;
}

<p>这是基准大小。</p>
<p id="fstline">这是 1 倍大小文字。</p>
<p id="secline">这是 1.5 倍大小文字。</p>
<p id="thrdline">这是 3 倍大小文字。</p>
```

效果如图 5-3 所示。

这是基准大小。

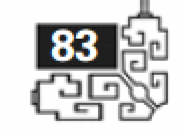
这是1倍大小文字。

这是1.5倍大小文字。

这是3倍大小文字。

图 5-3　p 元素的 font-size 属性值是相对于 body 的 font-size 属性值的

body 元素的字体大小为 small，所以第一个 p 元素中的文字大小也为 small。在图 5-2 中，1 倍大小文字与父元素 body 的文字大小值 small 相当，其余两行文字的大小分别是该基准值的 1.5 倍和 3 倍。



使用 em 作为单位的数值在小数点后面可以拥有三个有效位，另外通过 em 单位可以达到外部元素根据内部元素大小动态变化的效果。

提示：若开发人员没有设定基准值，那么 1em 表示的就是浏览器默认的一个单位的字符大小。大多数浏览器默认字体大小为 16px，因此在未给定基准值的情况下设置 font-size 属性为 1em 时，文字大小就是 16px。

ex 表示相对于字符“x”的高度。这个值会随着 x 的字体大小和字体族的不同而不同。如图 5-4 所示是在 font-size 属性为 25px 时，各个字体的 x 字符的高度值。

x in Arial (13px)

x in Courier New (11px)

x in Verdana (14px)

x in Tahoma (14px)

x in Times New Roman (12px)

x in Georgia (12px)

图 5-4　不同字体的 x 字符的高度(数值标注在后面的括号中)

在下面的示例中，两个段落将分别使用两种不同的字体，其 x 字符的高度不同，因此使用 ex 单位所产生的效果也会有所不同：

```
p{
    width:200px;
    margin:0 10px;
}
p#para1{
    font-family:Tahoma;
    height:20ex;
    float:left;
    background-color:#FFCC99;
}
p#para2{
    font-family:"Times New Roman";
    height:20ex;
    float:left;
    background-color:#FFCC99;
}

<p id="para1">Matrix: A situation or surrounding substance within which
something else originates, develops, or is contained.</p>
<p id="para2">Matrix: A situation or surrounding substance within which
something else originates, develops, or is contained.</p>
```

最终效果如图 5-5 所示(Firefox 浏览器)。

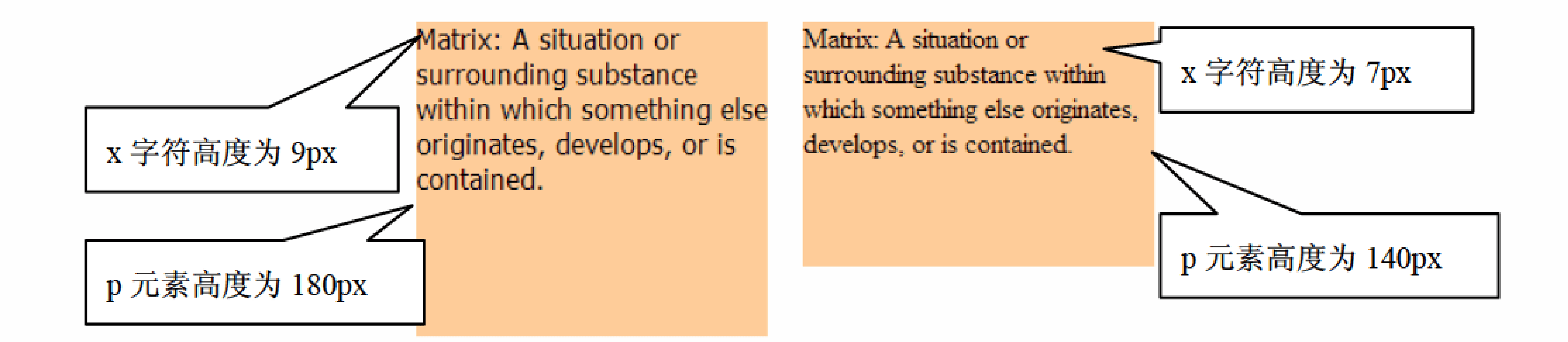


图 5-5　单位为 ex 的属性值是字符 x 高度的相对值(Firefox 浏览器)

第一个段落使用 Arial 字体，字符 x 的高度为 9 个像素，因此第一个 p 元素的高度为 20×9px =180px，第二个段落使用 Times New Roman 字体，字符 x 的高度为 7 个像素，因此该 p 元素的高度为 20×7px =140px。尽管两个 p 元素的高度均是 20ex，但由于字符 x 的高度不同，导致了 p 元素的高度也不相同。

最后一个表示相对长度的单位就是像素了。像素是相对与显示设备的分辨率。常见的显示器分辨率有 1024×768、1280×1024、1280×800 等。一般情况下，当页面需要精确控制其大小时，都使用像素作为其元素的单位。有些站点的元素会随着浏览器分辨率的变化而变化，因而可能会使用 em 或百分比一类的单位。

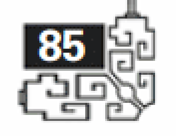
有些读者可能不太理解，为什么将像素视为相对单位呢？对于每个显示设备来说，像素的大小是固定不变的，但是在多个尺寸不同却拥有相同分辨率的设备之间，一个像素的大小就是变化的。也就是说一个边长为 100px 的正方形区域显示在尺寸不同的设备上时，实际的大小是有差异的。尽管如此，在大部分情况下，你仍然可将 px 视为一个绝对单位。

5.3　小　　结

每条有效的 CSS 规则至少包含一条声明，而声明中必须包含属性和属性值。CSS 正是通过给属性赋予不同的属性值才使得(X)HTML 代码能呈现出千变万化的样式。值和单位是 CSS 的核心内容之一，掌握它们对进一步使用好 CSS 起着相当重要的作用。

CSS 中的值类型包括整数、实数、长度、百分数、URI 和 URL、颜色、字符串。一些属性要求属性值(长度值)带有单位，CSS 中的单位分为表示绝对长度和相对长度两大类。绝对单位一般用于打印机等设备，相对长度通常用于显示设备，其中 em 和 px 较常使用。

下一章将向读者介绍 CSS 中的层叠和继承的概念。



第 6 章　层叠和继承

CSS 中处处存在继承，它非常重要，既给开发人员带来方便也带来麻烦。一方面它减少了样式表代码的数量，能达到事半功倍的效果；另一方面它很容易被错误地理解和使用，导致发生混乱，所以从开始学习 CSS 时就要注意这些问题。

本章将向读者介绍 CSS 中继承的原理，以及如何利用继承提高开发效率。接着会介绍 @import 规则的含义和用途、选择符的确定度以及层叠样式表中层叠的具体含义。

本章主要内容

- *什么是继承*
- *继承的作用和局限*
- *使用指定继承*
- *@import 规则*
- *层叠样式表中层叠的含义*

6.1　继　　承

6.1.1　什么是继承

简单地说，CSS 中的继承就是指某个(X)HTML 元素拥有的某些样式会传递给它的所有后代元素。比如一般情况下页面中的 p 元素是 body 的后代，所以 body 元素所拥有的部分样式也会被 p 元素继承下来。继承给开发人员节省了大量的时间，否则页面所有的元素都要设置一遍重复的样式信息。下面这个示例就利用了 CSS 的继承性质。

(X)HTML 代码：

```
<h1>Adobe Dreamweaver 简介</h1>
<p>
<strong>Adobe Dreamweaver</strong>是构建 Web 站点和应用程序的专业工具。它将可视布局工具、应用程序开发功能和代码编辑支持组合在一起，其功能强大，使得各个层次的开发人员和设计人员都能够快速创建界面吸引人的基于标准的网站和 Web 应用程序。从对基于 CSS 的设计的支持到手工编码功能，Dreamweaver 提供了专业人员在一个集成、高效的环境中所需的工具。</p>
```

CSS 规则只给 body 元素指定样式：

```
body{
    font-family:Tahoma, Arial, "宋体";
    color:#666666;
}
```

页面中的 h1 和 p 元素是 body 的后代元素，p 元素中的 strong 元素也是 body 的后代元素，所以页面所有元素的字体族和颜色属性都会继承祖先元素 body 的样式，结果如图 6-1 所示。

Adobe Dreamweaver 简介

Adobe Dreamweaver是构建Web站点和应用程序的专业工具。它将可视布局工具、应用程序开发功能和代码编辑支持组合在一起，其功能强大，使得各个层次的开发人员和设计人员都能够快速创建界面吸引人的基于标准的网站和Web应用程序。从对基于CSS的设计的支持到手工编码功能，Dreamweaver提供了专业人员在一个集成、高效的环境中所需的工具。

图 6-1　继承产生的效果

6.1.2　利用继承

继承给开发人员带来了极大的方便，减少了冗余的 CSS 样式代码。就像 6.1.1 节的示例所展示的那样，通过给 body 元素指定字体族属性，页面所有元素均会继承这个属性值，因而不必为每个元素单独编写字体族属性的样式代码。

你还可以利用继承性给页面某一部分区域指定样式。比如，页面设计人员通常把 div 元素作为容器，把页面划分成几个区域(如导航区、主要内容区、次要内容区和底脚区等)，区域内所包含的元素均是 div 的后代。通过给该 div 元素指定样式就能达到指定该区域内所有元素样式的目的。例如下面的示例就通过指定 div 元素字体族属性、字号大小和颜色等，使得其内部多个文字段落的样式达到统一。最终效果如图 6-2 所示。

小晖子工作室网站 Copyright©2005-2006 版权所有
E-mail:huistd@163.com
闽ICP备06016999号

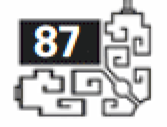
图 6-2　利用继承将样式赋给子元素，减少冗余样式代码

(X)HTML 代码：

```
<div id="footer">
    <div id="copyright">
        <p>小晖子工作室网站 Copyright&copy;2005-2006 版权所有</p>
        <p>E-mail:huistd@163.com</p>
        <p>闽 ICP 备 06016999 号</p>
    </div>
</div>
```

CSS 代码：

```
div#footer{
    font-family:Tahoma, Arial, "宋体";
    font-size:12px;
    text-align:center;
    color:#FFFFFF;
    background-color:#3F4C6B;
}
p{
    margin:0;
}
```



6.1.3 所有的规则都能继承吗

继承并不是想象的那样，后代元素会继承其祖先元素的所有规则。而是只有部分 CSS 规则会被其后代元素所继承。就拿边框(border，详见第 7 章)属性来说，父元素的边框样式并不会传递给它的后代元素，这也很合理，否则当你想给某个区域的 div 元素添加上一个边框时，该 div 元素所有后代元素周围均会有个边框。

假如我们在 6.1.1 小节的示例中给 body 元素添加一个黑色 1 像素宽的实线边框：

```
body{
    font-family:Tahoma, Arial, "宋体";
    color:#666666;
    border:1px solid black;
}
```

由于边框属性不被其后代元素继承，所以结果会如图 6-3 所示，只有 body 元素周围存在边框。

Adobe Dreamweaver 简介

Adobe Dreamweaver是构建Web站点和应用程序的专业工具。它将可视布局工具、应用程序开发功能和代码编辑支持组合在一起，其功能强大，使得各个层次的开发人员和设计人员都能够快速创建界面吸引人的基于标准的网站和Web应用程序。从对基于CSS的设计的支持到手工编码功能，Dreamweaver提供了专业人员在一个集成、高效的环境中所需的工具。

图 6-3 body 元素的 border 样式不会被其后代元素继承

若是所有属性均被后代元素继承，那么就会出现如图 6-4 所示的效果。

Adobe Dreamweaver 简介

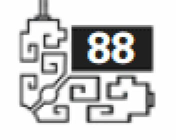
Adobe Dreamweaver是构建Web站点和应用程序的专业工具。它将可视布局工具、应用程序开发功能和代码编辑支持组合在一起，其功能强大，使得各个层次的开发人员和设计人员都能够快速创建界面吸引人的基于标准的网站和Web应用程序。从对基于CSS的设计的支持到手工编码功能，Dreamweaver提供了专业人员在一个集成、高效的环境中所需的工具。

图 6-4 假如所有属性均能被后代元素继承，页面上会到处都是边框

那么到底哪些属性支持继承，哪些不支持呢？请看下面三条规则：

- 影响元素位置，边界，背景和边框的属性不会被继承。
- 浏览器会使用它们自己的继承样式来格式化某些元素，比如标题(h1~h6)文字显示为粗体且字体较大，链接文字(a 元素)的颜色为蓝色等。这些元素相应的属性也不会继承它们祖先元素的属性。
- 当样式继承存在冲突时，采取就近原则。后代元素会继承距离自己最近的祖先元素的相应属性。

最后我们通过一个实例来进一步理解以上三条规则的内容。



(X)HTML 代码：

```
<div id="container">
    <h1>最新产品</h1>
    <h2>Web Professional</h2>
    <p>Web Professional 是一款<strong>功能完备的可视化的 Web 开发软件</strong>，
    它能快速地创建优美的 Web 页面，提供了强大的网站管理功能。欲获详情请点击<a
    href="webpro.html">此处</a>。</p>
    <h2>Web Express</h2>
    <p>Web Express 是一款<strong>轻巧的可视化 Web 开发工具</strong>，使用它可快速
    创建您所需要的 Web 页面。欲获详情请点击<a href="webexpr.html">此处</a>。</p>
</div>
```

给 body 元素添加如下样式：

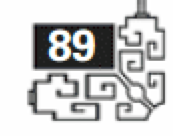
```
body{
    font-family:Tahoma, Arial, "宋体";
    font-size:0.7em;
    color:#A74500;
    text-align:center;
}
```

规则规定了字体族、字体大小和颜色，并要求文字水平居中，从图 6-5 可以看出 body 的后代元素 p 完全继承了这些规则。strong 元素并没有独自居中。而 h1 和 h2 元素只继承了字体族、颜色和文字居中的属性，它内部文字大小不受影响。链接文字只继承了字体族和字体大小的属性，文字颜色不受影响。

图 6-5　后代元素 h1、h2 和 a 不会全部继承其祖先元素的属性

接下来我们给 div 和 p 元素添加如下样式：

```
div#container{
    border:1px solid gray;
    width:260px;
    padding:1em;
    text-align:left;
}
p{
    margin-left:2em;
}
```



即给 div 元素添加 1 个像素宽的实线灰色边框，宽度固定为 260 个像素，填充大小为 1 个单位(前面讲过，这里 1 个单位就是指自身文字的 1 倍大小)。文字居左对齐。根据就近原则，div 内的元素将继承 div 元素居左对齐的属性，因为 div 比 body 元素更靠近这些元素，而 div 元素自己将居中显示在浏览器中。最后给 p 元素设定 2 个单位的左边距，由于边距属性不会被后代元素继承，所以 p 内的 strong 元素并没有这 2 个单位的边距。最终效果如图 6-6 所示。

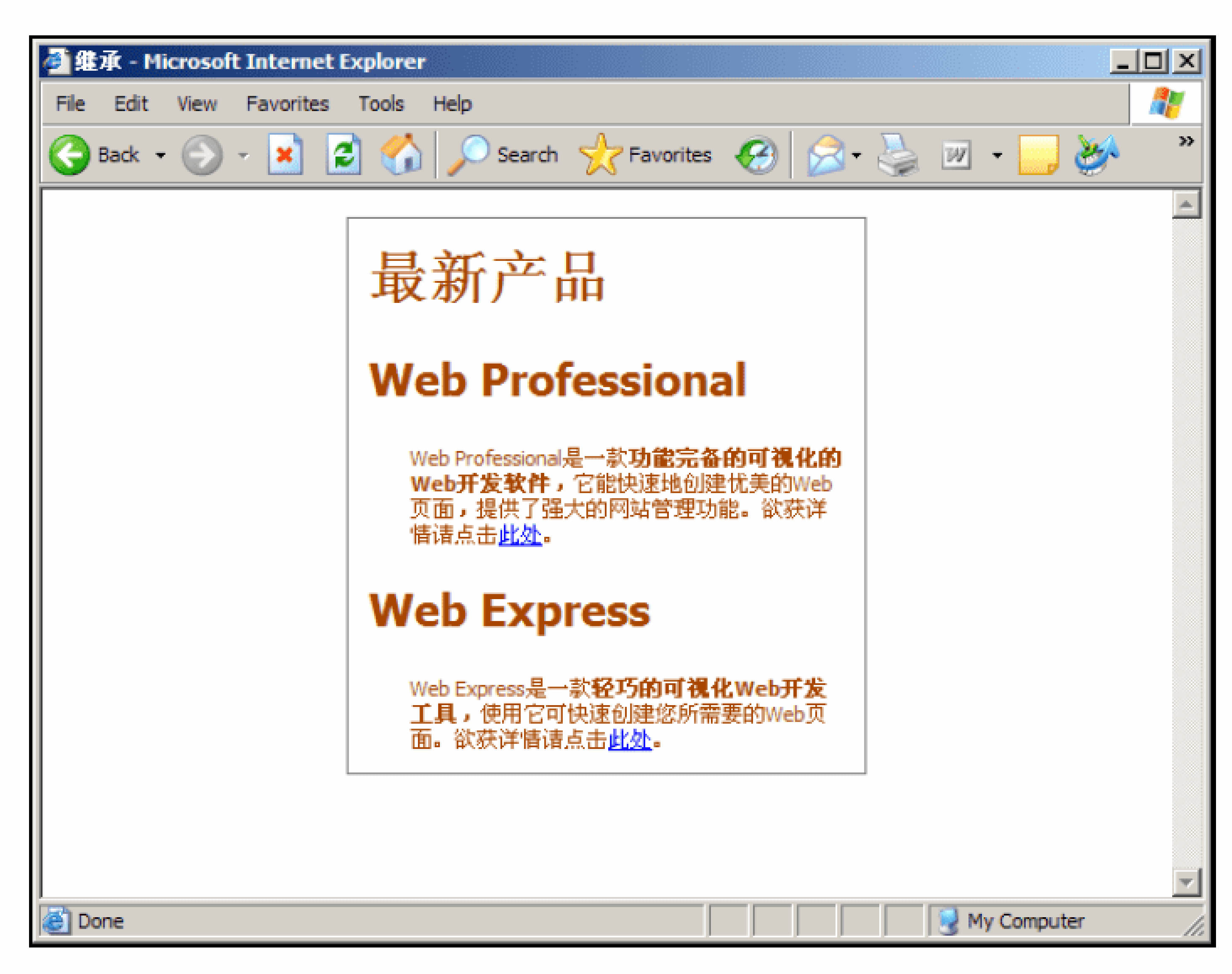


图 6-6　div 元素位于浏览器中央，但内部文字则向左对齐。p 元素的边距不会被其子元素 strong 继承

注意： 声明“text-align:center;”在 Firefox 浏览器中对 div 元素并不起作用，你可以使用声明“text-align:-moz-center;”来实现居中，但这时在 IE 中又无法居中。一般的解决方法是使用 margin 属性，给需要居中的元素添加两条声明：“margin-left:auto;”和“margin-right:auto;”，这样居中效果在 IE 和 Firefox 中都可以正确显示了。

6.1.4　inherit 指定继承

CSS 中的每个属性都有一个特定值“inherit”，其含义就是指定继承父元素的相应属性值。使用 inherit 一方面在代码中能显式地表明要继承于父元素的样式属性，另一方面也使子元素继承了那些不会被自动继承的属性。假如设计者要使 ID 为 menu 的 div 元素有 2px 的黑色边框，且具有 5px 的填充，同时想让其子 div 元素也具有同样的样式，则可以编写如下规则(注意，IE 浏览器无法正确显示)：

```
div#menu{
    border:2px solid black;
    padding:5px;
    background-color:#909090;
}
```

```
div#menu div{
    border:inherit;
    padding:inherit;
    background-color:#DEDEDE;
}
```

图 6-7　使用 inherit 显式继承其父元素的 border 和 padding 属性(Firefox 浏览器)

从图 6-7 中可以看出内部的 div 元素继承了父元素的 border 和 padding 属性。

6.2　@import 规则

CSS 中的@import 规则允许你向(X)HTML 文档或另一个外部样式表文档中添加外部样式。通过这种方式你可以添加附加样式到相应的文档中。实际上@import 规则的设计初衷并不是要在(X)HTML 文档中使用，而是要提供一种将一个或多个样式表导入到主外部样式表的方法。

@import 规则的使用方式如下所示：

```
@import url("stylesheet1.css");
@import url("stylesheet2.css");
```

其中 url()部分可以省略不写，若要严格遵循 XHTML 规范，引号不能省略，末尾的分号不能丢。以下 4 种写法是等效的(注意 Firefox 浏览器不支持最后一种写法)：

```
@import url("stylesheet1.css");
@import url(stylesheet1.css);
@import "stylesheet1.css";
@import stylesheet1.css;
```

6.2.1　@import 规则的用途

一些低版本的浏览器不支持@import 规则，它们会忽略这条规则，因此导入的样式也会被忽略掉。另外，IE 浏览器支持条件注释，也可以结合@import 规则和条件注释添加一些针对特定浏览器的样式，比如下面这段取自微软英文首页的代码就利用@import 规则和条件注释给版本低于 IE7 的浏览器添加特定的样式：

```
<style type="text/css">
    @import url('http://i2.microsoft.com/shared/core/1/css/core.css');
</style>
<!--[if lt IE 7]>
<style>
    @import url('http://i2.microsoft.com/shared/core/1/css/ie6.css');
</style>
```

```
<![endif]-->
<style type="text/css">
    @import url('http://i2.microsoft.com/shared/core/1/css/lsp.css');
</style>
```

6.2.2 @import 规则的使用

下面这段代码是将外部样式表 external.css 导入到(X)HTML 文档中，注意代码位于<head>和</head>之间：

```
<style type="text/css">
@import url("external.css");
<style>
```

添加到样式表文档中就更简单了，只需一行代码：

```
@import url("external.css");
```

需要注意的是，不论@import 规则出现在(X)HTML 文档的 style 元素中还是出现在 CSS 文档中，它都必须写在所有其他 CSS 规则之前，否则@import 规则导入的样式信息不起任何作用。比如下面这种写法就使得 external.css 中的任何样式属性都不起作用：

```
<style type="text/css">
body{
    background-color:white;
}
@import url("external.css");
</style>
```

但是这样写是可行的：

```
<style type="text/css">
body{
    background-color:white;
}
</style>
<style type="text/css">
@import url("external.css");
</style>
```

6.3 层叠的含义

CSS 全称为 Cascading Style Sheet，第一个单词 Cascading 直译为层叠或级联，这个概念看来相当重要，以致于该语言都要由它来命名，那么这个层叠的具体含义是什么呢？

页面所使用的样式规则可能有不同的来源，其中有些规则由 Web 设计者提供，有些由用户提供，有些又是由浏览器提供(比如浏览器中字体颜色默认为黑色)。这三种来源不同的样式



可能会存在重复、相互冲突的内容，它们会影响页面中的同一个元素。那页面最终的样式取决于哪一个呢？这就要根据层叠来规定。

CSS 中的层叠是一种规则，它分配给每一条样式规则一个权值。当若干规则一起使用时，权值最高的样式将会起作用。在 CSS 中与权值密切相关的一个概念就是确定度(Specificity)，下面就来介绍它。

6.3.1　确定度

通过 CSS 选择符，我们可以有多种方式去选定(X)HTML 元素。有时可能会出现多个样式直接应用到同一元素上的情况。比如页面中存在这样一段(X)HTML 代码：

```
<p id="para1" class="normal">...</p>
```

p 元素的 id 为 para1，class 为 normal，那么下面任何一条规则都会应用到这个元素上：

```
p {color:red;}
body p {color:silver;}
p#para1 {color:green;}
p.normal {color:blue;}
```

显然，p 元素中文字的颜色只可能是一种，那么它到底取决于哪一条规则呢？这是由选择符的确定度来决定的。每种 CSS 选择符都被指定了一个确定度，当若干存在冲突样式属性的规则应用到同一元素时，确定度最高的那一条将会产生效果。

选择符的确定度由 4 位逗号相隔的数字组成，形式为 0,0,0,0。我们分别用 a、b、c 和 d 代表这 4 位数。这 4 位数的确定方法如下：

- 若样式由元素的 style 属性确定而不通过选择符，将 a 置 1，否则置 0。
- 计算 id 选择符个数，赋值给 b。
- 计算属性选择符或伪类个数，赋值给 c。
- 计算类型选择符或伪元素个数，赋值给 d。

只要通过比较确定度的大小就可以判定出优先级最高的规则，比较方式是从最左端的 a 位开始逐位比较，数值较大的确定度就高，若该位相等则比较下一位，以此类推。现在你可以试着计算下列规则的确定度，看看是否和其后标注的数值一致：

```
* {}                /* a=0 b=0 c=0 d=0 -> 确定度 = 0,0,0,0 */
p {}                /* a=0 b=0 c=0 d=1 -> 确定度 = 0,0,0,1 */
p:first-line {}     /* a=0 b=0 c=0 d=2 -> 确定度 = 0,0,0,2 */
ul li {}            /* a=0 b=0 c=0 d=2 -> 确定度 = 0,0,0,2 */
ul ol+li {}         /* a=0 b=0 c=0 d=3 -> 确定度 = 0,0,0,3 */
h1 + *[lang=zh]{}   /* a=0 b=0 c=1 d=1 -> 确定度 = 0,0,1,1 */
ul ol li.red {}     /* a=0 b=0 c=1 d=3 -> 确定度 = 0,0,1,3 */
li.red.level {}     /* a=0 b=0 c=2 d=1 -> 确定度 = 0,0,2,1 */
#header {}          /* a=0 b=1 c=0 d=0 -> 确定度 = 0,1,0,0 */
style=""            /* a=1 b=0 c=0 d=0 -> 确定度 = 1,0,0,0 */
```

从确定度的计算规则来看，内联样式的优先级最高，且在一般情况下，选择符的特定性越

高，它的优先级也越高。我们现在来计算一下本节开始提到的 4 条规则的确定度，看看到底哪一条规则的优先级最高：

```
p {color:red;}              /* a=0 b=0 c=0 d=1 -> 确定度 = 0,0,0,1 */
body p {color:silver;}      /* a=0 b=0 c=0 d=2 -> 确定度 = 0,0,0,2 */
p#para1 {color:green;}      /* a=0 b=1 c=0 d=1 -> 确定度 = 0,1,0,1 */
p.normal {color:blue;}      /* a=0 b=0 c=1 d=1 -> 确定度 = 0,0,1,1 */
```

ID 选择符的确定度最高，因此 p 元素中文字颜色为绿色。

6.3.2 !important 声明

有时，你并不希望浏览器按照选择符的确定度来决定某些样式属性的优先级。CSS 2.1 提供了一个“!important”声明(由叹号和单词 important 组成)，拥有该声明的属性比其他任何没有该声明的属性的优先级都要高。使用时将“!important”声明放置在样式声明的属性值与表示结束的分号之间，比如：

```
p {color: red !important;}
```

请看下面的示例。

(X)HTML 代码：

```
<h2 id="articleTitle" class="highlighted">使用!important 声明</h2>
```

CSS 代码：

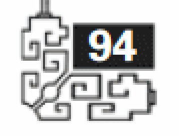
```
h2{                    /* 确定度 = 0,0,0,1 */
    color:red !important;
    font-weight:bold;
}
#articleTitle{         /* 确定度 = 0,1,0,0 */
    color:green;
    background-color:silver;
}
.highlighted{          /* 确定度 = 0,0,1,0 */
    background-color:yellow !important;
}
```

如图 6-8 所示为以上代码的效果。尽管选择符 h2 与.highlighted 的确定度都比#articleTitle 低，但是“!important”声明使得元素内部的文字为红色、背景为黄色。

使用!important声明

图 6-8　!important 声明的属性值优先级最高

若不使用“!important”进行声明，根据选择符的确定度文字应为绿色，背景应为银灰色。



注意：一些 Web 设计者认为“!important”声明破坏了 CSS 选择符原有的确定度规则，影响了 CSS 的结构性，因此不推荐在设计者的 CSS 中使用。由于一般情况下，设计者的 CSS 样式要比用户的优先级高，所以“!important”声明经常用在用户的样式表中，用来覆盖设计者的某些样式(有关设计者和用户的样式表请参考下一小节的内容)。

6.3.3　层叠顺序

前文提到，层叠是若干规则的集合，用来决定元素该如何应用样式属性。它规定浏览器在 CSS 规则发生冲突时该如何处理应用到同一元素上的多个样式。CSS 规范规定了层叠顺序。

(1) 寻找到所有应用到元素上的样式声明。

(2) 通过声明的重要度(也称权重，看声明是否标有“!important”)与来源排序，按照有限度从高到低的顺序为：

- 用户(User)重要声明
- 设计者(Author)重要声明
- 设计者一般声明
- 用户一般声明
- 浏览器默认声明

(3) 通过选择符的确定度给重要度和来源一致的样式规则排序：确定度较高的样式规则先于确定度较低的样式规则。

(4) 最后，通过规则先后顺序排序：如果两条规则的权重、来源和确定度一致，后声明的样式先于之前声明的样式。被导入进来的样式规则要先于执行导入操作的样式规则。

为了更进一步理解以上规则是如何生效的，我们通过具体示例来解释三条排序规则。

1. 通过声明的权重和来源排序

第 2 条层叠顺序规则规定：若两条规则应用到同一元素，标有!important 的声明优先。比如下面两条规则会使 p 元素内的文字为银灰色而不是红色：

```
p {color: silver !important;}
p {color: red;}
```

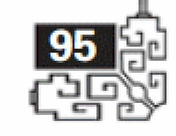
此外，该层叠顺序规则还将样式表的来源考虑在内，请看如下规则：

```
p {color: silver;}  /* 来自设计者的样式 */
p {color: red;}     /* 来自用户的样式 */
```

两条样式规则分别来自设计者和用户，这时来自设计者的样式优先。但是如果声明均标有!important 声明，则用户的样式优先，比如：

```
p {color: silver !important;}   /* 来自设计者的样式 */
p {color: red !important;}      /* 来自用户的样式 */
```

这也说明了用户对页面样式的确定有着最高的优先权。假如一个视力有障碍的人感觉某些页面的字体偏小，则可以在用户样式表中为页面设置一个较大的字体号，并添加“!important”



声明。这样所有页面都会按照这个规则进行显示。

最后一点，不管是设计者的样式还是用户的样式，它们都优先于浏览器的默认样式。

提示：设计者样式表是指网页开发人员编写的样式表，它和页面文件一起放在 Web 服务器上。用户样式表则由访问者自己编写，存储在本地磁盘上，通过浏览器的相关设置可以将样式应用到页面中。在 IE 浏览器中，你可以通过 Internet 选项里的可访问性按钮选择用户样式表文件。在 Firefox 中，可以向 userContent.css 文件添加自定义样式。Opera 浏览器的偏好高级设定中可以指定用户样式表。

2. 通过选择符的确定度排序

我们已经在 6.3.1 小节介绍过确定度的概念以及计算方法。第 3 条层叠顺序规则规定：若两条规则应用到同一元素且它们的权重和来源一致，则确定度较高的样式规则先于确定度较低的样式规则。请看如下示例。

CSS 代码：

```
h1.decorated{
    color:red;
}
h1#newsTitle{
    color:blue;
}
h1{
    color:black;
}
```

(X)HTML 代码：

```
<h1 id="newsTitle" class="decorated">W3C 推出 CSS 新规范</h1>
```

这三条样式规则的确定度依次为 0,0,1,1、0,1,0,1 和 0,0,0,1，显然此标题文字为蓝色。

3. 通过规则先后顺序排序

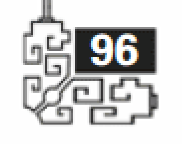
最后一条规则规定：如果两条样式规则的重要度、来源和确定度均一致，则后声明的规则要优先于先声明的规则。比如下面这段声明会使段落文字为绿色：

```
p {color:red;}
p {color:green;}
```

6.4 小　　结

层叠概念是 CSS 的关键内容，层叠顺序规定了浏览器该如何处理存在冲突的样式声明。其中还涉及到继承机制、导入外部样式表、选择符的确定度等概念。

页面中子元素可自动继承其祖先元素的部分样式属性，并采取就近原则。对于那些不能自动继承的属性可使用 inherit 进行指定继承。



@import 规则允许开发人员将外部样式表导入到当前样式表文档或(X)HTML 文档中。在导入到样式表文档时，@import 规则必须写在所有其他 CSS 规则之前才能生效。使用该规则开发人员还可以根据浏览器版本动态调用不同的样式表文档，从而达到良好的兼容性。

层叠顺序首先根据声明的重要度和来源排序：作者样式表优于用户样式表，用户重要样式优于作者重要样式，不论怎样它们都优于浏览器默认的样式。标有!important 的样式声明视为重要声明。其次，层叠顺序根据选择符的确定度来排序，确定度较高的样式规则优于确定度较低的样式规则。最后，根据样式出现的次序来排序，后定义的样式优于先定义的样式。

下一章将介绍有关 CSS 盒模型的内容。

第 7 章　盒　模　型

盒模型是 CSS 中较为重要的核心概念之一，它是使用 CSS 控制页面元素外观样式的基础。只有充分理解盒模型概念才能进一步掌握 CSS 的正确使用方法。本章从介绍盒模型概念开始，依次介绍其所涉及的边距、边框和填充等概念，接着会介绍如何控制元素的宽度和高度，最后介绍 overflow 属性与 display 属性的含义和作用。

本章主要内容

- CSS 盒模型的概念
- 什么是边距、边框和填充
- 宽度和高度
- overflow 属性
- display 属性

7.1　盒模型概述

(X)HTML 提供了各种各样的元素，用来表示不同的内容，例如 p 元素表示段落，h 元素表示标题，img 元素表示一幅图像，……。当浏览器显示这些元素的时候，把每一个元素都视为一个方形的盒子(Box)。CSS 盒模型(Box Model)描述了根据文档树生成的一系列矩形区域。浏览器会分析下载的网页，并根据其中的内容在内存中构造一颗文档树，然后将文档树中可视的内容绘制在浏览器中，形成我们看到的网页。

盒模型涉及内容(Content)、边距(Margin)、填充(Padding)和边框(Border)四个概念，如图 7-1 所示为盒模型的示意图。

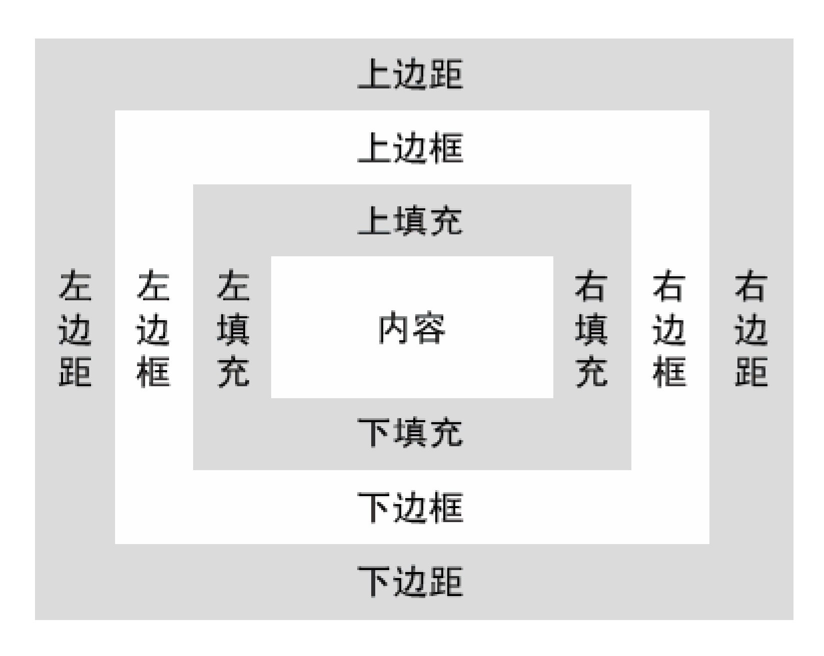


图 7-1　盒模型示意图

提示： 有些书中也称边距为外边距，称填充为内边距。

图 7-1 绘制了盒模型的各个部分及相互之间的位置关系。位于盒模型中心位置的是元素内容，比如 p 元素中的文字，img 元素中的图像，该区域的大小正好可以包含元素中所有的内容。与内容紧挨着的是填充区域，填充可使内容和边框之间产生一定的空间，填充外侧是元素的边框，边框之外是边距区域，边距使得本元素与周围元素之间产生一定的空间。

默认状态下，不同的(X)HTML 元素会包含不同的盒模型属性，比如 p 元素默认情况下就会有上下边距的存在。图 7-2 展示了一个实际的例子。

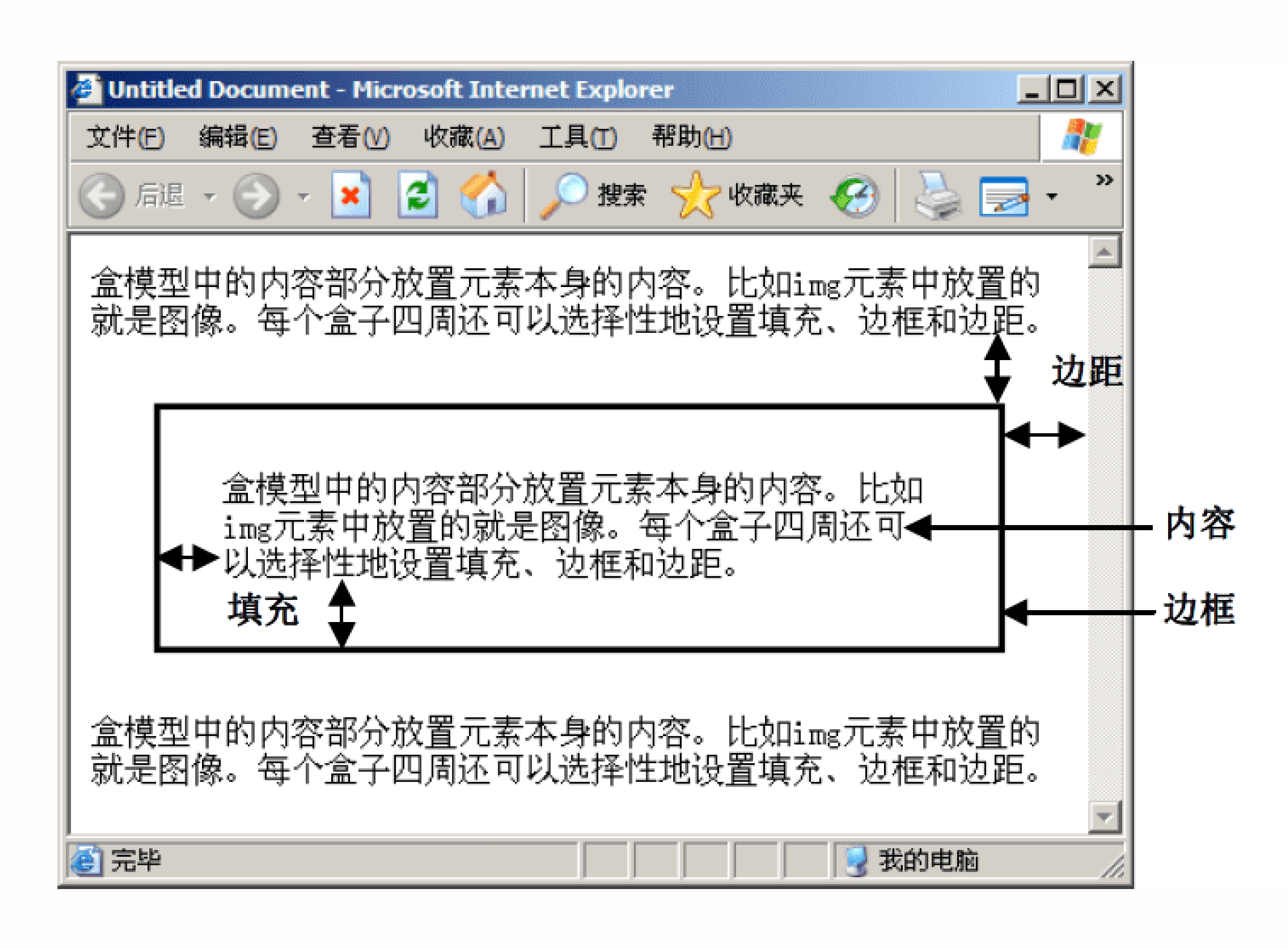


图 7-2　浏览器中盒模型的各个部分

7.2　边　　框

使用边框可以给元素四周添加一个矩形的线框，由图 7-1 的盒模型示意图可知，边框位于填充和边距之间。CSS 针对边框提供了三类样式属性，分别控制边框样式风格、边框颜色和边框的粗细程度。另外，四个方向的边框样式是独立的，可以分别进行设置。

7.2.1　边框样式风格

CSS 为设计者提供了相当丰富的边框样式，使用如下属性就可以分别对上、右、下、左四个方向的边框设置不同的样式：

- border-top-style
- border-right-style
- border-bottom-style
- border-left-style

边框风格属性可使用的属性值及其产生的效果如表 7-1 所示。

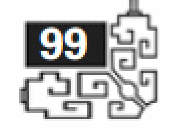


表 7-1 边框风格属性及其效果

边框风格属性值	产生的效果
dashed	虚线边框
dotted	点状边框
double	双层边框
groove	凹槽效果，与 ridge 效果相反
hidden	隐藏边框
inset	下凹效果，与上凸效果相反
none	不设置边框
outset	上凸效果，与下凹效果相反
ridge	边框中心隆起，与 groove 效果相反
solid	实心边框

各种风格在 IE 浏览器中的显示如图 7-3 所示。

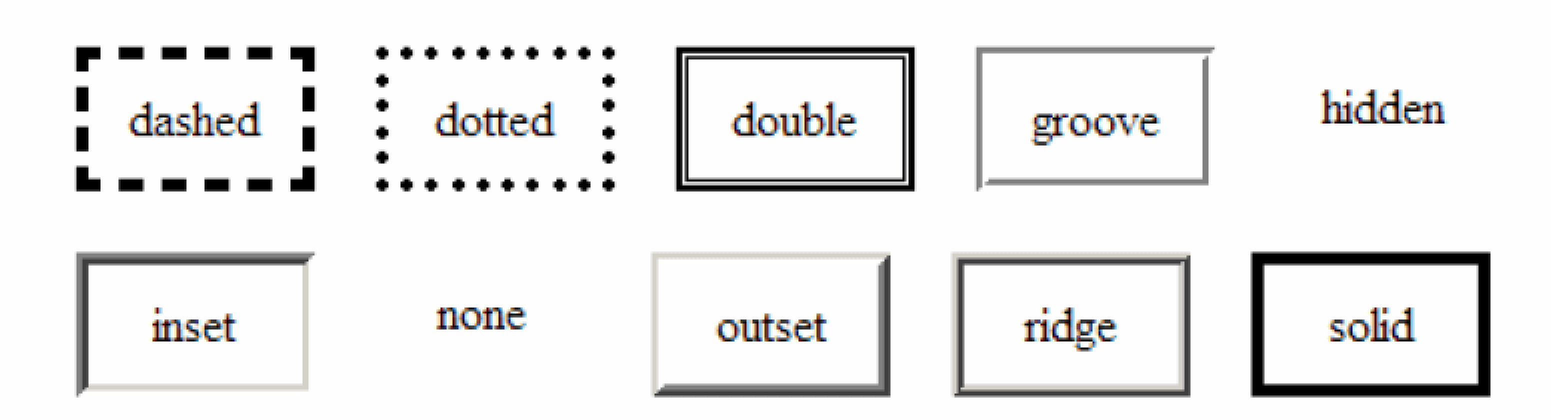


图 7-3 各种边框风格在 IE 中的显示效果

不同的浏览器对于边框风格的显示效果存在细微的差异，比如 Firefox 中的双层边框，它的两层边框粗细程度一致，而 IE 则有差别。如图 7-4 所示为各种风格的边框在 Firefox 浏览器中的显示效果，读者可以仔细比较它们之间的不同。

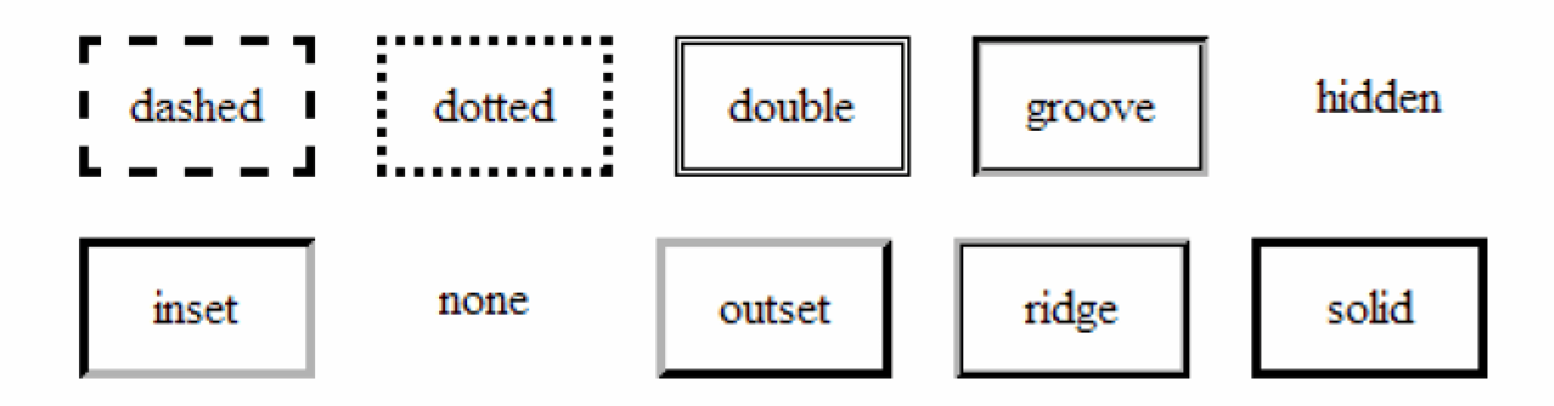
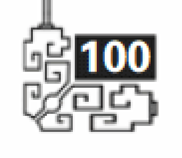


图 7-4 各种边框风格在 Firefox 中的显示效果

7.2.2 边框颜色和粗细程度

除了边框的风格样式外，边框的颜色和粗细程度也可由 CSS 进行控制。下面 4 个属性分别控制上、右、下、左 4 个边框的颜色：border-top-color、border-right-color、border-bottom-color 和 border-left-color。属性值为 CSS 中合法颜色值以及表示颜色的关键字。假如没有给边框指定



颜色，其颜色依 color 属性而定。

与颜色类似，控制边框粗细程度的 4 个属性为：border-top-width、border-right-width、border-bottom-width 和 border-left-width。这些属性除了可以使用 CSS 的长度值和百分数外，还能使用如下 3 个关键字：thin、medium 和 thick。

例如，以下样式代码会产生如图 7-5 所示效果：

```
p#p1{
    text-align:center;
    border-top-style:dashed;
    border-right-style:dotted;
    border-bottom-style:double;
    border-left-style:dotted;
    border-top-color:red;
    border-right-color:#4DE354;
    border-bottom-color:rgb(100, 200, 50);
    border-left-color:blue;
    border-top-width:medium;
    border-right-width:1px;
    border-bottom-width:0.6em;
    border-left-width:thick;
}

<p id="p1">边框属性示例</p>
```



图 7-5　边框属性示例

注意：在图 7-5 中，p 元素左右边框样式都是 dotted，但是在 IE 浏览器中，当边框宽度设成 1px 后，其样式转变为 dashed 了。

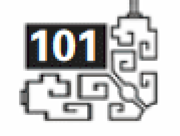
7.2.3　边框样式缩写形式

从 7.2.2 小节的示例中可以看到，每个边框都要用独立的 CSS 属性进行设定，要想控制一个元素所有边框的所有样式，就需要编写大量的代码，好在 CSS 给我们提供了相关样式的缩写形式。

border-style 属性可同时指定 4 个边框的样式，该属性后面可以跟随 1 至 4 个边框样式属性值。含义如下：

```
border-style:四个边框的风格;
border-style:上下边框的风格 左右边框的风格;
border-style:上边框的风格 右左边框的风格 下边框的风格;
border-style:上边框的风格 右边框的风格 下边框的风格 左边框的风格;
```

图 7-6 展示了以下 4 条 CSS 声明所产生的样式效果：



```
border-style:solid;
border-style:solid dotted;
border-style:solid double ridge;
border-style:dashed solid dotted double;
```


图 7-6　border-style 属性的用法

提示： border-style 属性后面跟随 4 个属性值时，其影响边框的顺序可记为英文单词：TRouBLe(麻烦，问题)，其中的大写字母正好是 top、right、bottom 和 left 的首字母。也可记为从 top 开始的顺时针方向。

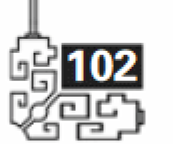
类似地，颜色和宽度都有缩写形式：border-color 和 border-width。属性值的个数和含义同 border-style 属性。这样，7.2.2 小节中的示例代码就可重写为如下形式：

```
p#p1{
    text-align:center;
    border-style:dashed dotted double;
    border-color:red #4DE354 rgb(100, 200, 50) blue;
    border-width:medium 1px 0.6em thick;
}
```

边框样式属性的缩写形式除了可以按类别划分外还可以按照方向划分，你可以分别控制上、右、下、左四个方向的边框的所有样式，它们对应的属性分别为：border-top、border-right、border-bottom 和 border-left。该属性后面跟随宽度值、边框风格和颜色，三个属性值的顺序是任意的。另外，如果元素的四个边框样式一致，还可以使用更简便的 border 属性。

以下代码将产生如图 7-7 所示的效果：

```
p#p1{
    border-top:dotted 4px  red;
    border-right:solid 2px yellow;
    border-bottom:dotted 4px red;
    border-left:solid 2px yellow;
}
p#p2{
    border:1em solid #EDDE34;
}
```



```
p#p3{
    border:2px dashed #777;
    border-left:6px solid #79E283;
}

你可以使用如下四个属性同时给每个边框添加属性：
<p id="p1">border-top border-right border-bottom border-left</p>
你还可以使用如下属性同时给所有边框添加属性：
<p id="p2">border</p>
使用了 border 属性后，还可以给某个或某几个边框添加属性，比如：
<p id="p3">border: dashed #777 2px;<br />border-left:solid #79E283 6px;</p>
```

你可以使用如下四个属性同时给每个边框添加属性：

border-top border-right border-bottom border-left

你还可以使用如下属性同时给所有边框添加属性：

border

使用了border属性后，还可以给某个或某几个边框添加属性，比如：

border: dashed #777 2px;
border-left:solid #79E283 6px;

图 7-7　border 属性示例

7.3　填充和边距

盒模型中的填充和边距都可以用来给元素内容周围增加空间。但是填充和边距位于盒模型中的不同部分，因此它们的用法和作用也不相同。

7.3.1　填充

填充是指元素中内容与边框之间的区域。CSS 提供了如下 4 个属性分别控制元素的上、右、下、左四个方向的填充：padding-top、padding-right、padding-bottom 和 padding-left。属性值类型为长度值和百分数。padding 为填充的缩写属性，其属性值的个数和每个属性值所代表的方向与 border-style 属性是一致的。

请看如下示例：

```
p{
    border:1px solid gray;
    padding:10px 30px;
}
```

```
img{
    border:1px dashed black;
    padding:16px;
}

<p>填充是元素内容与边距之间的空白区域，CSS 提供了 padding-top、padding-right、padding-bottom 和 padding-left 属性分别控制上、右、下、左四个方向的填充。padding 属性为缩写形式，后可跟 1 至 4 个属性值。</p>
<img src="images/flower.jpg" />
```

效果如图 7-8 所示。

填充是元素内容与边距之间的空白区域，CSS提供了padding-top、padding-right、padding-bottom和padding-left属性分别控制上、右、下、左四个方向的填充。padding属性为缩写形式，后可跟1至4个属性值。



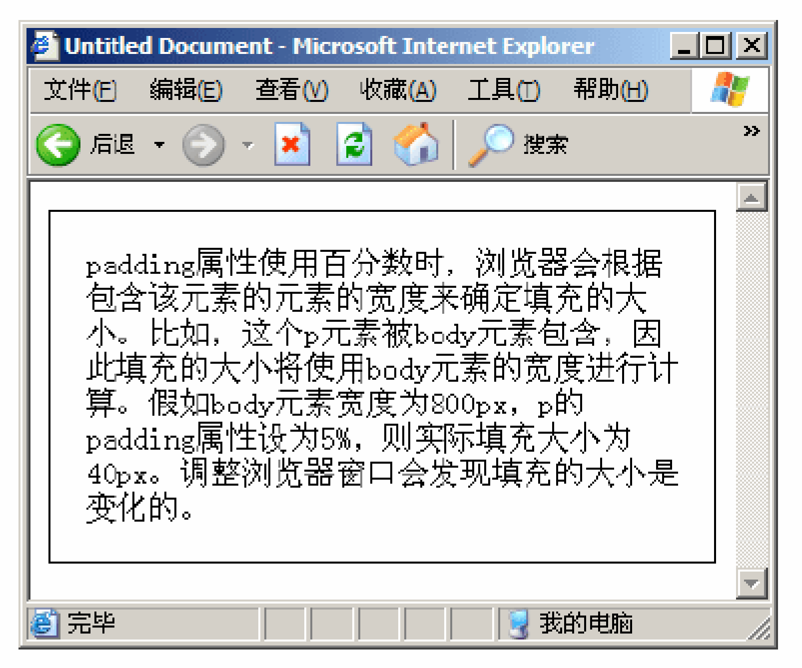
图 7-8 padding 属性的使用

padding 属性使用百分数时，浏览器会根据包含该元素的元素的宽度来确定填充的大小。例如：

```
p{
    padding:10%;
    border:1px solid black;
}
<p>padding 属性使用百分数时，浏览器会根据包含该元素的元素的宽度来确定填充的大小。比如，这个 p 元素被 body 元素包含，因此填充的大小将使用 body 元素的宽度进行计算。假如 body 元素宽度为 800px，p 的 padding 属性设为 5%，则实际填充大小为 40px。调整浏览器窗口会发现填充的大小是变化的。</p>
```

如图 7-9 所示为以上代码在不同大小的浏览器窗口中的效果，可以看到它们的填充是不相同的。





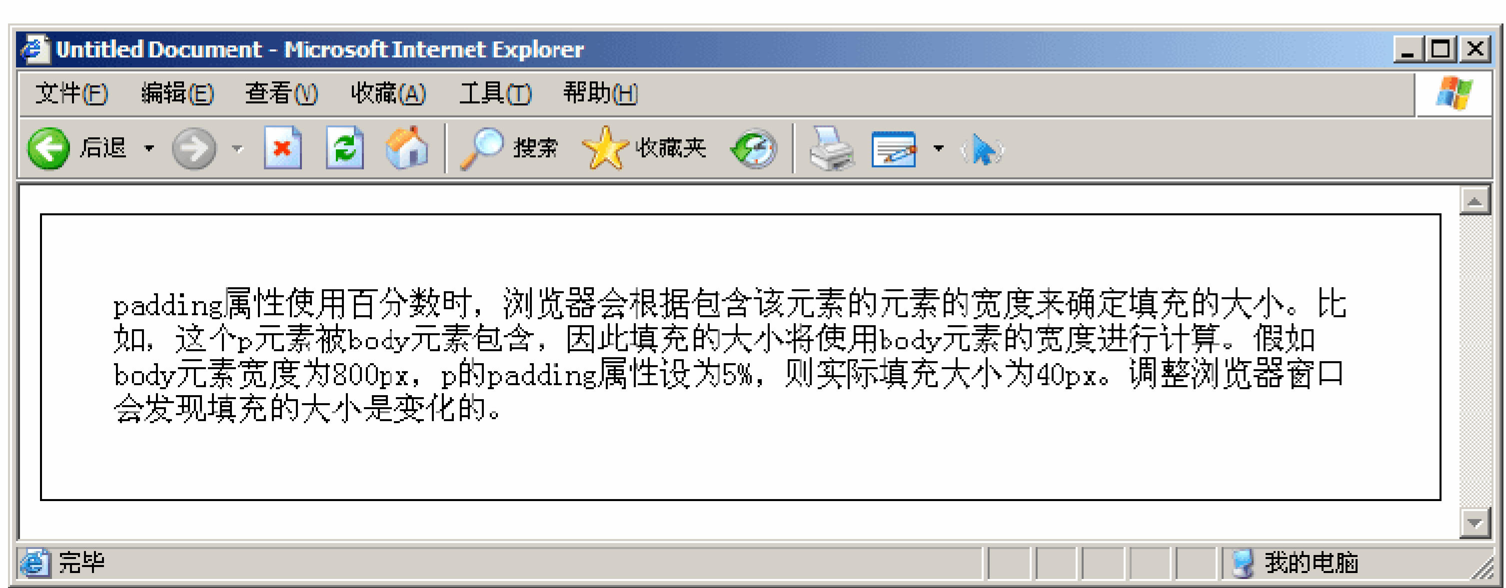


图 7-9　使用百分数作为 padding 属性值

7.3.2　边距

边距位于盒模型的最外侧，使用边距可以在相邻的多个元素之间产生空白区域。和填充一样，CSS 为每个方向的边距都提供了独立的属性，它们是 margin-top、margin-right、margin-bottom 和 margin-left，以及边距属性的缩写属性 margin，该属性所跟属性值的个数及其所代表的方向与 padding 属性相同。

用来表示段落的 p 元素在默认情况下就含有一定的边距，使得段落与其他元素之间产生一定的空间。请看下面的示例：

```
<p id="p1">边距区域位于边框的外侧，通常用来在多个相邻的元素之间添加空白区域。</p>
<p id="p2">默认情况下，许多(X)HTML 元素都包含一定的边距，比如 p 元素、body 元素。</p>
```

以上代码会产生如图 7-10 所示的效果。

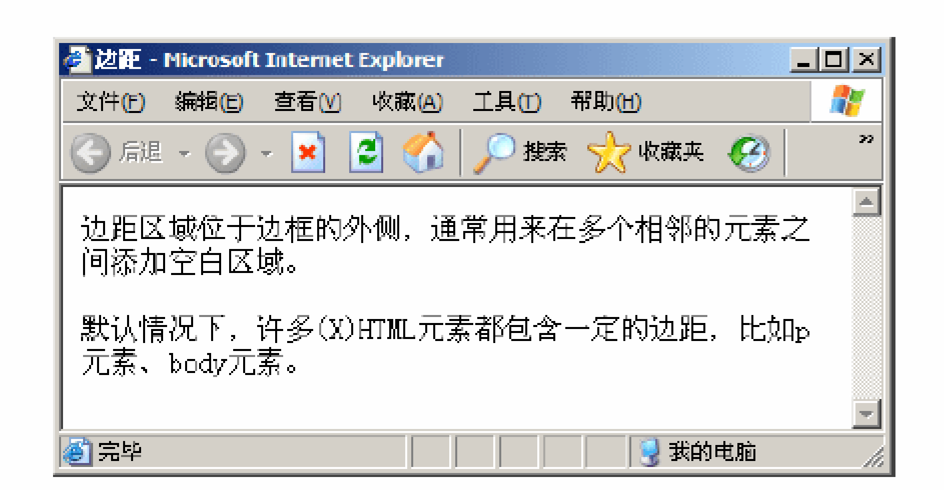
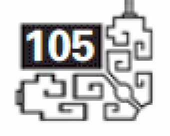


图 7-10　p 元素和 body 元素默认情况下存在边距



如果添加以下的样式，将 p 元素的 margin 属性设为 0，则会产生如图 7-11 所示的效果：

```
*{
    margin:0;
}
```

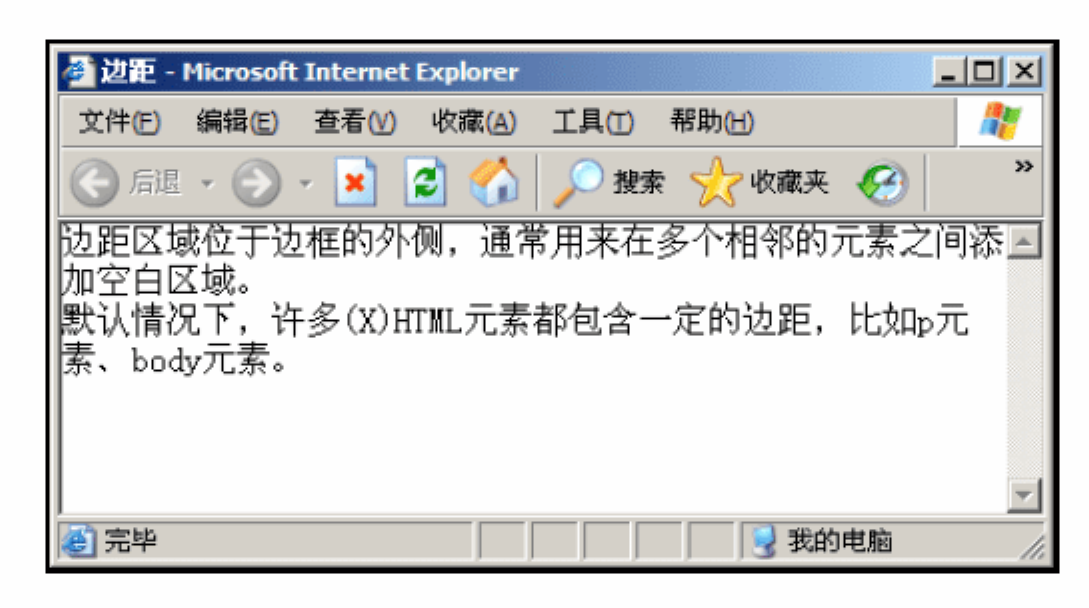


图 7-11　取消边距后的效果

对比图 7-10 和图 7-11 可以发现，文字会紧贴着浏览器窗口的边框，两个段落之间的距离也没有了，这都是 margin 属性的作用。

提示：边框、边距和填充亦可用在内联元素上，但是它们只在水平方向上产生正常的效果，而竖直方向上的效果与块级元素不同。

7.4　宽度和高度

前面说过，盒模型中内容区域的大小完全是由元素中内容多少决定的，比如 p 元素的内容区域会根据其文字的大小和多少等因素而定，而 img 元素的内容区域与其图像的大小一致。CSS 提供了两个属性可以改变盒模型中内容默认的大小：宽度属性 width 和高度属性 height。

7.4.1　width 和 height

1. width 属性

width 属性用来设置元素内容的宽度，属性值可以是长度值或百分数，它的默认值为 auto。请看下面的示例：

```
p#p1{
    width:230px;
}
p#p2{
    width:70%;
}
```



```
<p id="p1">盒模型中的内容区域为(X)HTML 元素中内容部分所占，默认情况下，该区域的大小完全由元素内容而定。使用 CSS 提供的 width 属性可以改变默认的宽度值。宽度值可以是长度值或者百分数。</p>
<p id="p2">盒模型中的内容区域为(X)HTML 元素中内容部分所占，默认情况下，该区域的大小完全由元素内容而定。使用 CSS 提供的 width 属性可以改变默认的宽度值。宽度值可以是长度值或者百分数。</p>
```

以上代码设定第一个段落宽度为 230px，第二个段落宽度为 70%，这个值是以该元素的父元素宽度为基础计算的，在这里是 body 元素。效果如图 7-12 所示。

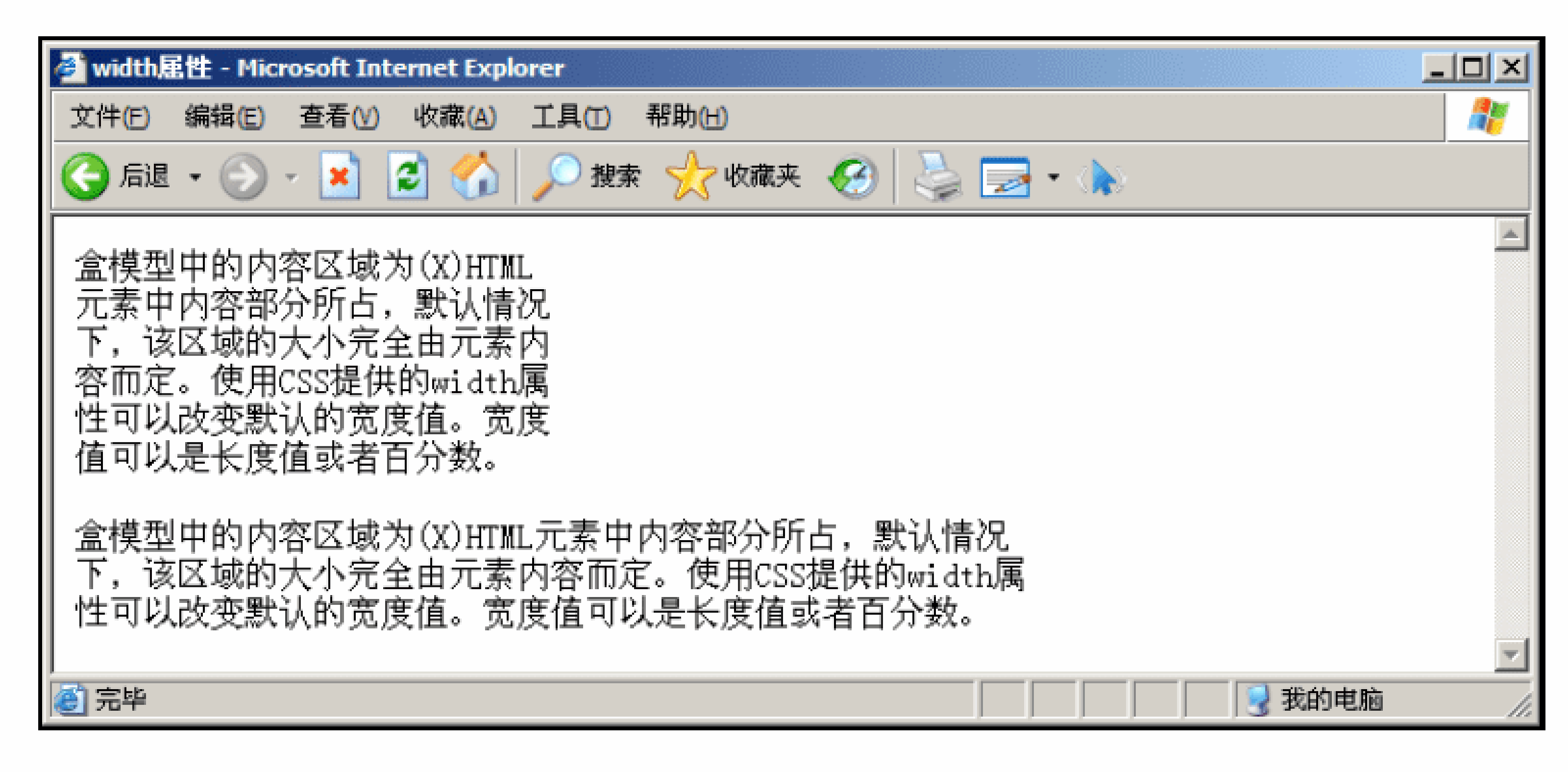


图 7-12　width 属性

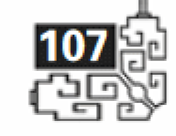
2. height 属性

和 width 属性类似，height 属性用来控制盒模型中内容区域的高度。属性值为长度值或百分数，默认值为 auto。

但是和 width 属性不同的是，当 height 值小于内容区域默认高度值时，元素内容仍旧按照原来的大小显示，也就是说元素内容不会被“截断”。请看如下示例：

```
<style type="text/css">
p#p1{
    width:400px;
    height:150px;
    background-color:#EEE;
}
p#p2{
    width:40%;
    height:40px;
    background-color:#EEE;
}

<p id="p1">盒模型中的内容区域为 HTML 元素中内容部分所占，默认情况下，该区域的大小完全由元素内容而定。使用 CSS 提供的 height 属性可以改变默认的宽度值。高度值可以是长度值或者百分数。</p>
<p id="p2">盒模型中的内容区域为 HTML 元素中内容部分所占，默认情况下，该区域的大小完全由元素内容而定。使用 CSS 提供的 height 属性可以改变默认的宽度值。高度值可以是长度值或者百分数。</p>
```



这里，除了给每个段落添加高度和宽度属性外，我们还用了一个 background-color 属性为元素添加背景颜色，后面还会讲到这个属性，这里使用为的是更清楚地分辨出元素的大小。图 7-13 展示了 IE6(左)和 IE7(右)的显示效果。

盒模型中的内容区域为HTML元素中内容部分所占，默认情况下，该区域的大小完全由元素内容而定。使用CSS提供的height属性可以改变默认的宽度值。高度值可以是长度值或者百分数。

盒模型中的内容区域为HTML元素中内容部分所占，默认情况下，该区域的大小完全由元素内容而定。使用CSS提供的height属性可以改变默认的宽度值。高度值可以是长度值或者百分数。

盒模型中的内容区域为HTML元素中内容部分所占，默认情况下，该区域的大小完全由元素内容而定。使用CSS提供的height属性可以改变默认的宽度值。高度值可以是长度值或者百分数。

盒模型中的内容区域为HTML元素中内容部分所占，默认情况下，该区域的大小完全由元素内容而定。使用CSS提供的height属性可以改变默认的宽度值。高度值可以是长度值或者百分数。

图 7-13　IE6(左)显示效果错误，IE7(右)的显示效果正确

在 IE6 中，盒模型的内容区域和元素内容区域一致，并没有严格按照 height 的属性值确定高度，而 IE7 则修正了这个错误(注意背景色覆盖的区域表示盒模型的内容区域)。Firefox 和 Opera 浏览器的显示是正确的。

当 height 属性值为百分数时，该元素的高度由其父元素高度和该百分比相乘而得，前提是父元素的高度不能是 auto，否则该父元素内所有后代元素的 height 属性值都将为 auto。请看下面的示例：

```
div#wrapper{
    width:100px;
    height:100px;
    background:gold;
}
div#inner{
    width:50%;
    height:50%;
    background:gray;
}

<div id="wrapper"><div id="inner"></div></div>
```

效果如图 7-14 所示。

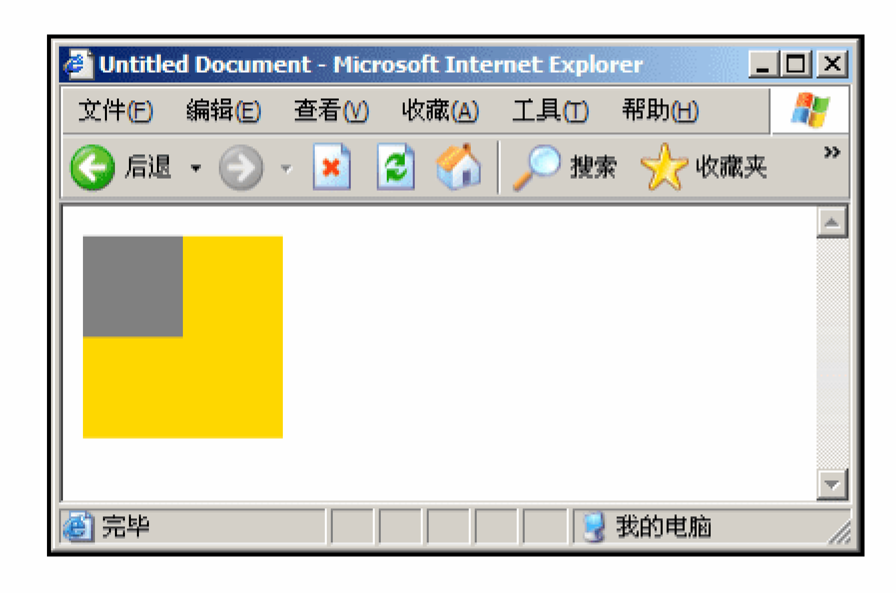
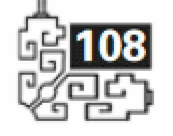


图 7-14　使用长度值的 height 属性



但如果将#wrapper 的 height 属性改为 70%，则会产生如图 7-15 所示效果。

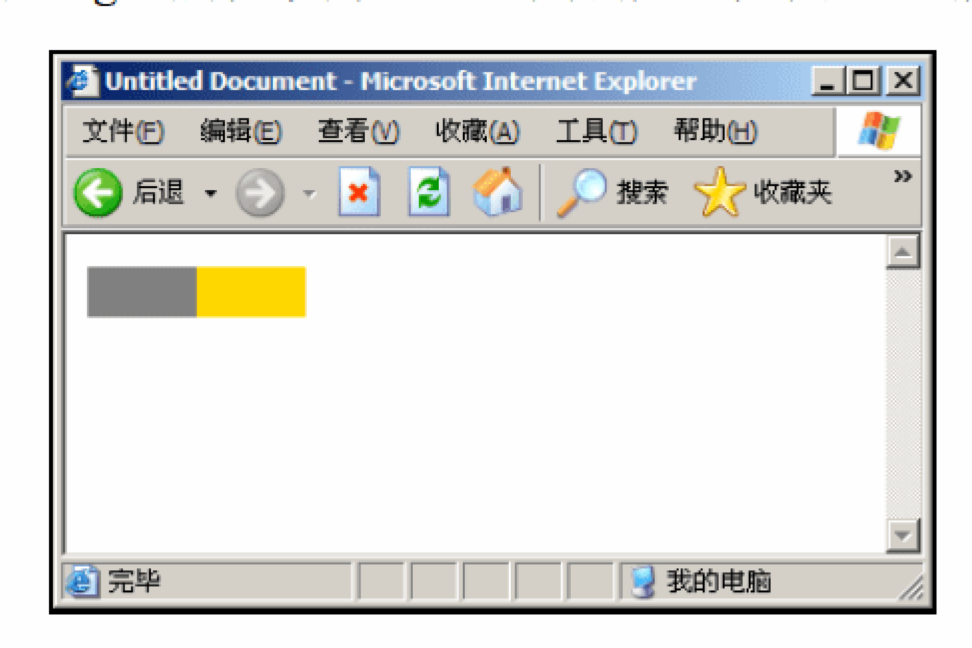


图 7-15　使用百分数的 height 属性

图 7-15 与我们设想的有些差异，#wrapper 的高度并不是浏览器窗口高度的 70%，里面的 div 元素也不是目前父元素高度的 50%。原因在于，#wrapper 的父元素为 body，而 body 的高度属性默认为 auto，其高度根据它所含内容多少而定，并不是浏览器窗口的高度。

如果给 body 添加一个确定的高度值，则会产生如图 7-16 所示效果：

```
body{
    margin:0;
    height:400px;
}
```

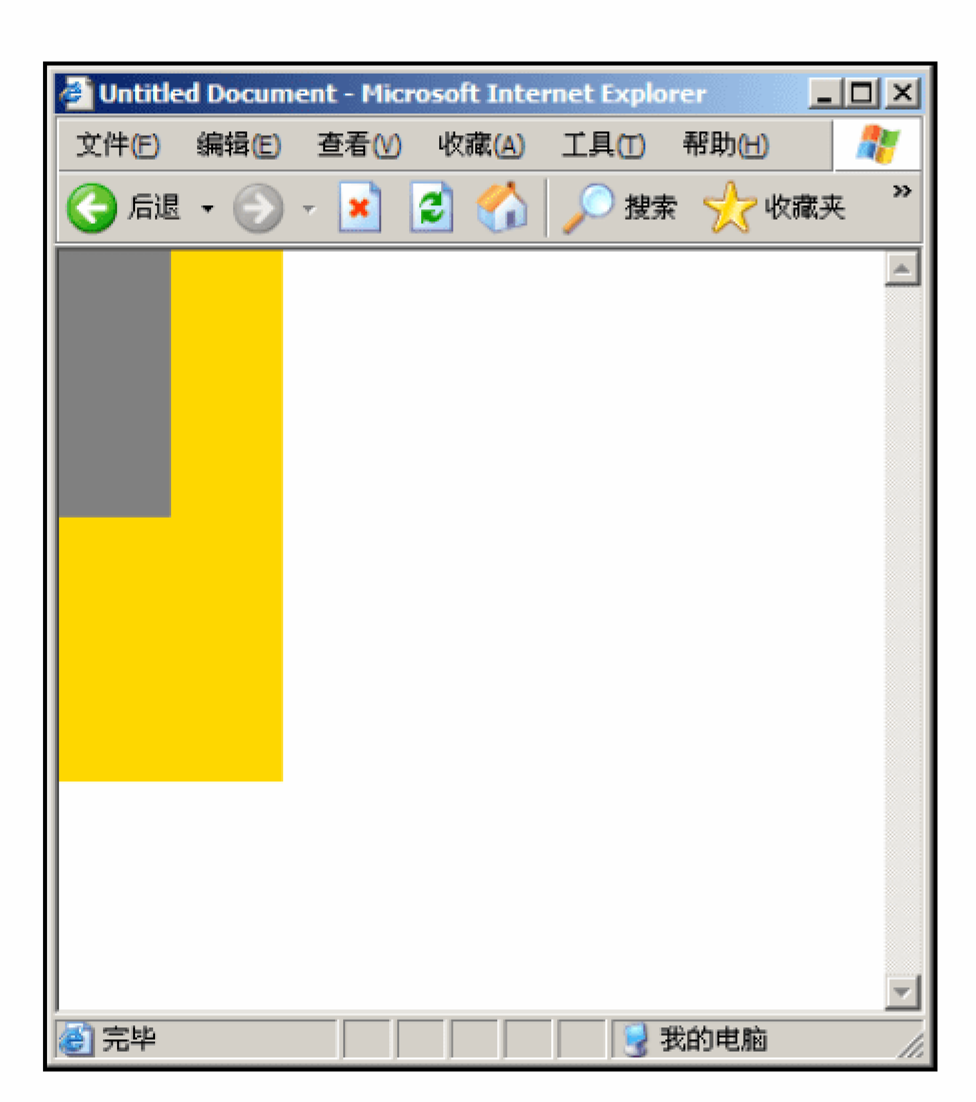
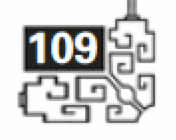


图 7-16　父元素高度确定后，后代元素高度的百分数才会起作用

从图 7-16 可以看出，#wrapper 的高度是 body 元素高度的 70%，即 280px，前提是给 body 元素设定了确定的 height 属性，而不是默认的 auto。

注意：如果将上例 body 元素的高度设为百分数，则只有在 IE6 中，div 元素的高度会按照百分数进行计算。读者可以亲自试一试。



前面说过，任何元素的 width 和 height 属性的默认值均为 auto，这时，元素宽度和高度的变化与元素本身的类型有关。比如 p 元素水平方向会占满整个空间，而竖直方向则依据其所包含内容的多少而定，就是块级元素的显示特征，和 p 元素相同的还有 div、h1~h6 元素等。

在下面的代码中，我们给 h1 和 p 元素指定了不同的背景色，以便能看出它们所占据空间的大小：

```
h1{
    background-color:red;
}
p{
    background-color:yellow;
}

<h1>auto 值的含义</h1>
<p>当块级元素的宽度和高度为 auto 时，宽度会占满整个水平方向上的空间，而高度会根据其所包含内容的多少而定。</p>
```

效果如图 7-17 所示。

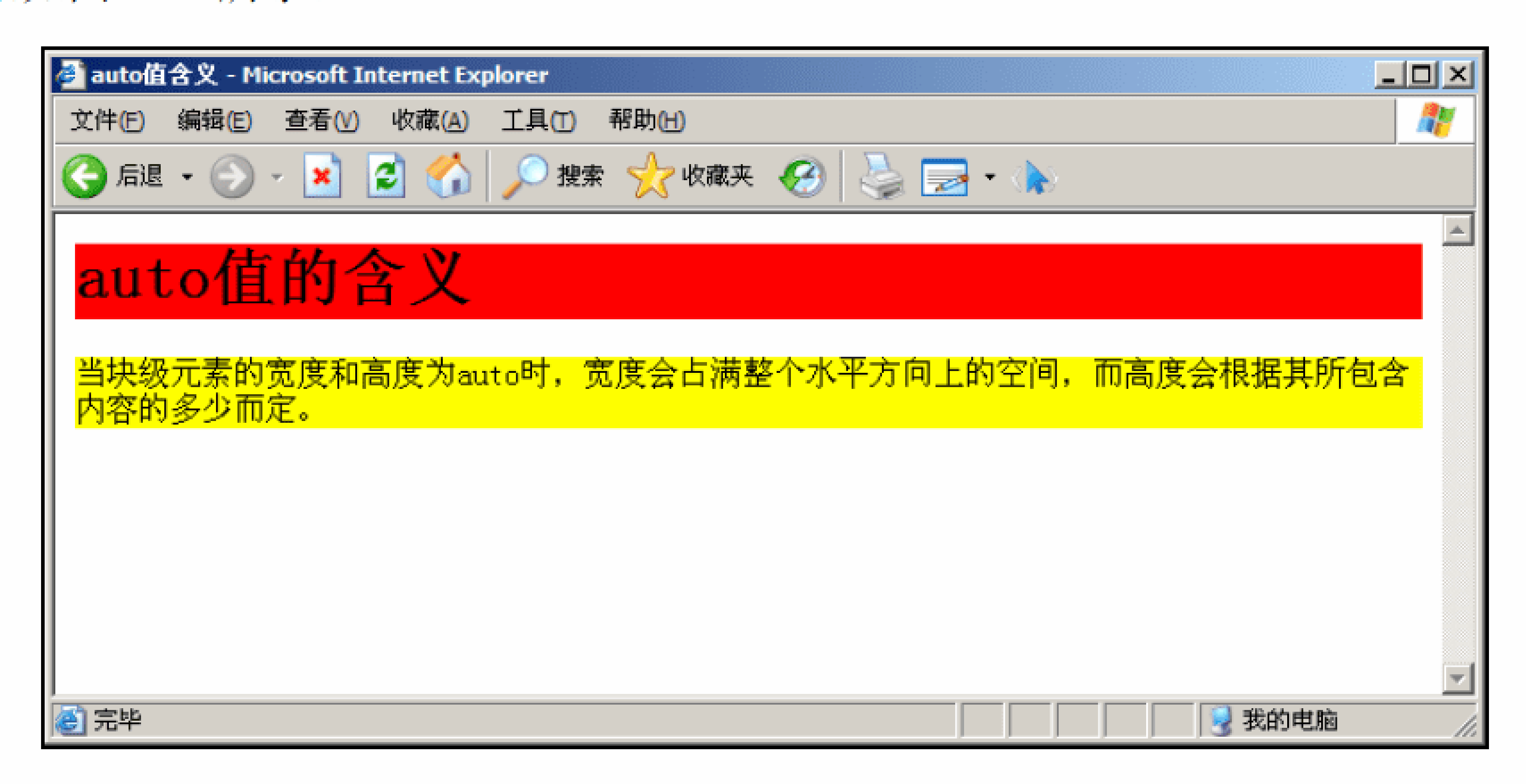
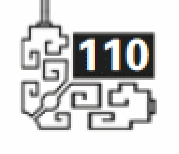


图 7-17　块级元素宽度和高度的变化

对于显示图像的 img 元素，auto 值意味着图像按照其原本的尺寸进行显示，而对于 table 元素，它的宽度和高度都根据其所包含的内容多少而变化。

7.4.2　最大值和最小值

有时候，我们需要某些元素在一定范围内随着父元素的大小变化，当超出这个范围时采用固定大小。CSS 提供的 min-width、max-width、min-height 和 max-height 就可以完成这个功能。比如页面中有个 p 元素，设计者希望该元素宽度能随着浏览器宽度的变化而定，以适应不同的分辨率，假如某用户的显示器分辨率为 1600×1200，那么浏览器最大化时的宽度将接近 1600px，这时我们不希望 p 元素宽度被拉伸至浏览器窗口的宽度，而是要求固定在某个确定值。这时就可以使用 CSS 提供的这些属性来实现这个功能。



1. min-width 和 max-width

min-width 和 max-width 属性分别设定元素的最小宽度值和最大宽度值，在下面的示例中，p 元素被设定了最小和最大宽度：

```
p{
    min-width:300px;
    max-width:600px;
    background-color:yellow;
}

<p>CSS 提供的 min-width 和 max-width 属性可设定元素宽度的最小值和最大值。在此范围内，元素的宽度会跟随父元素变化，若超出这个范围，元素宽度将保持不变。</p>
```

请看图 7-18 所示的效果。

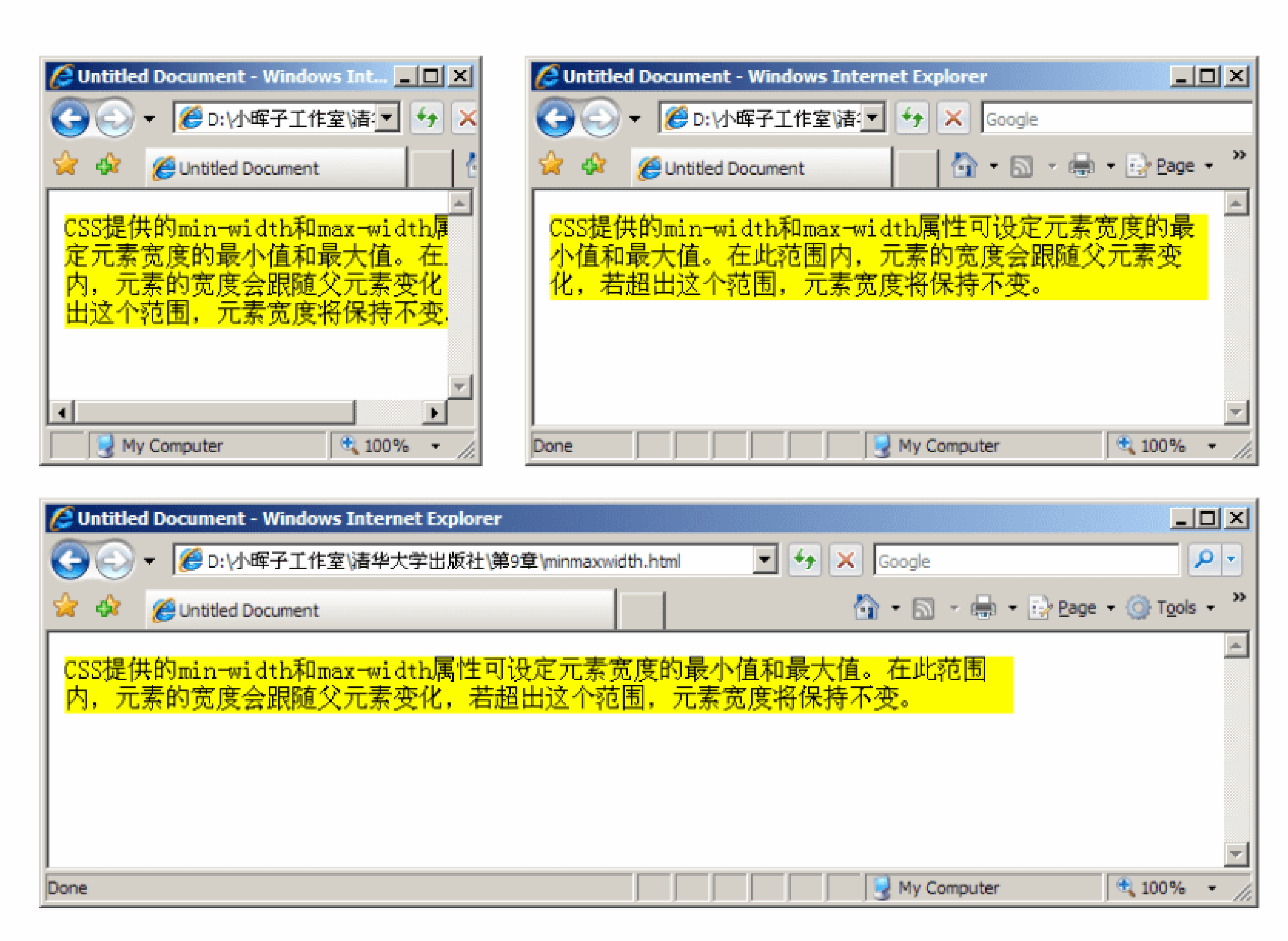
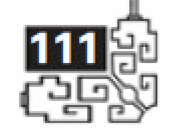


图 7-18　min-width 和 max-width 属性的使用(IE7)

从图 7-18 可以看出，当浏览器窗口宽度过小时，p 元素的宽度保持在 min-width 属性设定的 300px，当浏览器窗口宽度过大时，p 元素的宽度为 max-width 设定的 600px，在此范围内 p 元素宽度则会跟随浏览器窗口宽度的变化而变化。

2. min-height 和 max-height

min-height 和 max-height 属性分别设定元素的最小高度值和最大高度值，它们通常用于那些高度动态变化的元素，比如 p 元素的高度会根据其中文字的多少动态变化。假如我们又不希望该元素的高度变得过小或者过大，这时就可以使用 min-height 和 max-height 给元素限定其最小和最大的高度值。



请看如下示例：

```
p{
    width:350px;
    background-color:yellow;
    min-height:60px;
    max-height:100px;
}

<p>CSS 提供的 min-height 和 max-height 属性可分别设定元素的最小高度值和最大高度值。</p>
<p>CSS 提供的 min-height 和 max-height 属性可分别设定元素的最小高度值和最大高度值，它们
通常用于那些高度动态变化的元素，比如 p 元素的高度会根据其中文字的多少动态变化。假如我们又不
希望该元素的高度变得过小或者过大，这时就可以使用 min-height 和 max-height 给元素限定其最
小和最大的高度值。</p>
```

效果如图 7-19 所示。

CSS提供的min-height和max-height属性可分别设定元素的最小高度值和最大高度值。

CSS提供的min-height和max-height属性可分别设定元素的最小高度值和最大高度值，它们通常用于那些高度动态变化的元素，比如p元素的高度会根据其中文字的多少动态变化。假如我们又不希望该元素的高度变得过小或者过大，这时就可以使用min-height和max-height给元素限定其最小和最大的高度值。

图 7-19　min-height 和 max-height 属性的使用(IE7)

注意：本节介绍的 width 和 height 属性只适用于 p、div 一类的块级元素，对于 span、a 这样的内联元素则不起作用。有关块级元素和内联元素的知识请参考本书的第一篇。另外，本章 7.7 节将介绍与元素类型相关的 display 属性，读者可了解更多元素类型的知识。

宽度和高度的最大值和最小值在 IE6 中不起作用，针对这一问题可以使用 CSS 表达式来解决。

7.5　盒模型相关内容及高级主题

7.5.1　盒模型的维度

元素的 width 属性和 height 属性所设定的是元素内容区域的宽度和高度，由盒模型示意图可知，元素内容区域外侧还包含填充、边框和边距区域，当元素显示在页面中时，它的总体宽度和高度将由如下公式计算(盒模型维度示意图如图 7-20 所示)：

- 元素的宽度 = 内容宽度 + 左右填充宽度 + 左右边框宽度 + 左右边距宽度
- 元素的高度 = 内容高度 + 上下填充高度 + 上下边框高度 + 上下边距高度

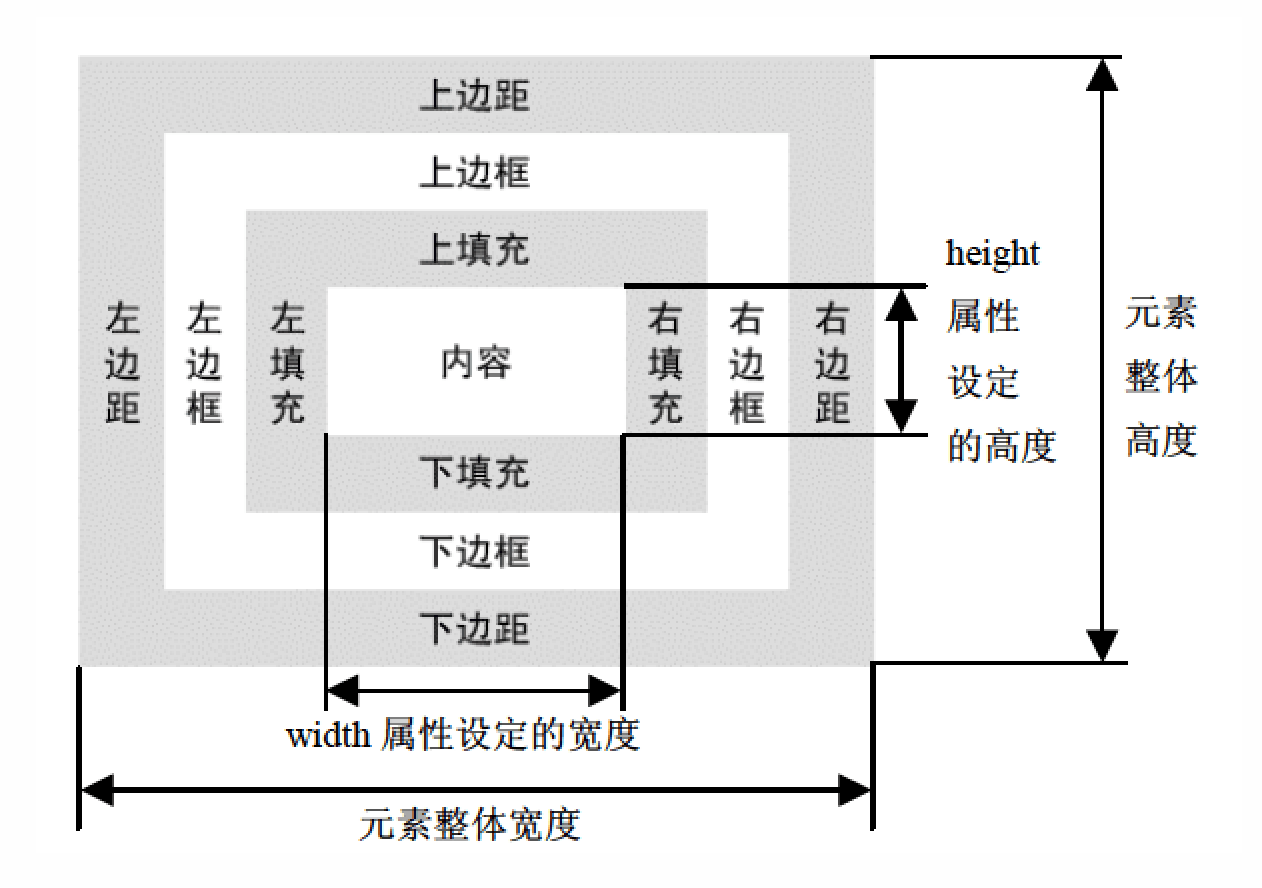


图 7-20　盒模型维度示意图

7.5.2　盒模型与背景

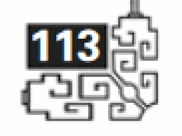
CSS 的 background 属性可用来设定元素背景，为其添加颜色或背景图像。背景属性会影响盒模型中的内容和填充两部分区域，而边框背景是由边框样式本身提供的，边距区域的背景则总是透明的。有关 background 属性的详细内容我们会在第 11 章进行介绍。

请看如下示例：

```
body{
    margin:0;
    background-color:silver;
}
div{
    height:50px;
    width:50px;
    margin:20px;
    padding:10px;
    border:10px solid black;
    background-color:yellow;
}

<div></div>
```

上述样式将 body 元素边距设为 0，并设定背景色为银灰色。div 元素内容的高度和宽度为 50px，边距为 20px，填充为 10px 再加上 10px 的黑色实线边框。从图 7-21 可以看出，div 边距部分透出了 body 元素的银灰色背景，边框部分为黑色，边框里面的填充和内容部分的背景色均为黄色。



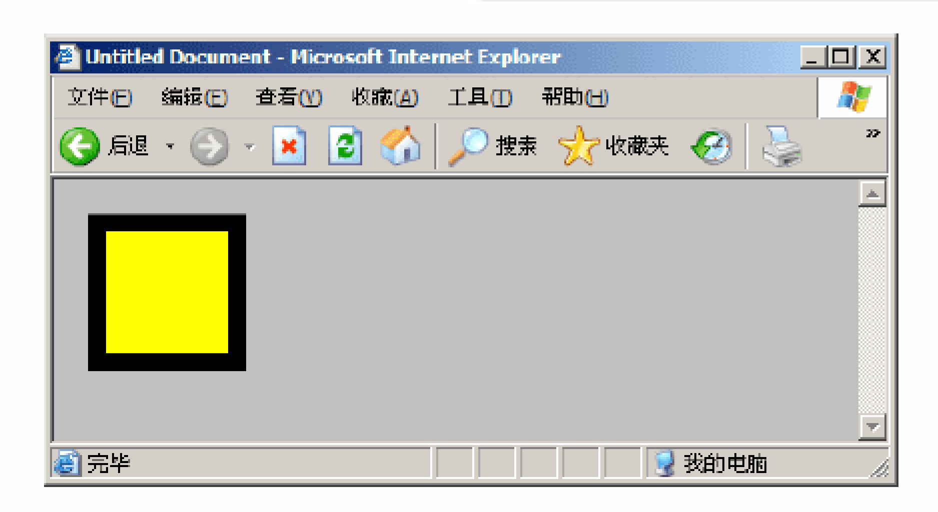


图 7-21　盒模型与背景

延伸：当元素边框风格为 dotted 或 dashed 时，边框中的空隙部分会显露出背景颜色或图像。也就是说，背景实际上画到了边框的位置，只不过位于边框的下面。以下这个链接展示了三维形式的 CSS 盒模型各个区域之间的关系：
http://www.hicksdesign.co.uk/journal/3d-css-box-model

7.5.3　边距重叠问题

边距重叠(Margin Collapsing)的意思就是当上下两个元素相邻时，二者之间的边距由它们的最大值而定，并不是二者边距之和。请看下面的示例：

```
body{
    margin:0;
}
div{
    width:100px;
    height:100px;
    border:2px solid black;
    background-color:yellow;
    margin:10px;
}

<div id="divTop"></div>
<div id="divBottom"></div>
```

我们给上下两个 div 元素添加 10px 的边距，但是在图 7-22 中，两个 div 元素之间的距离并不是 20px，而是 10px。边距发生了重叠。

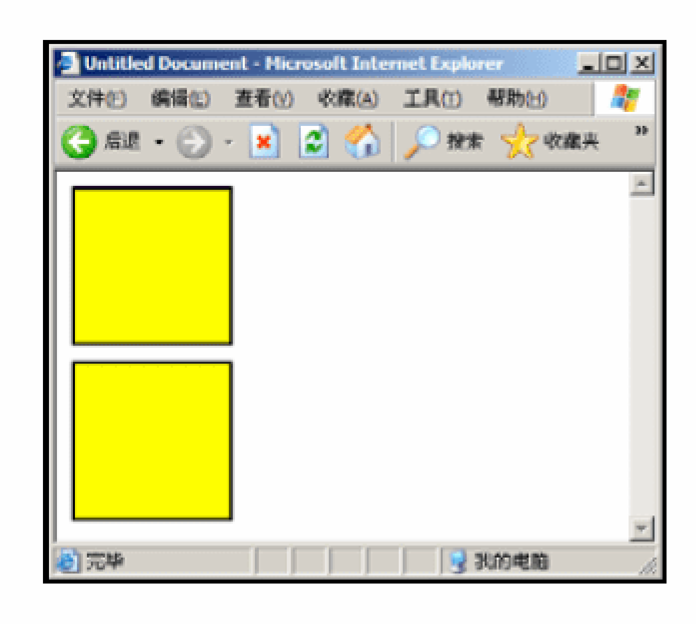
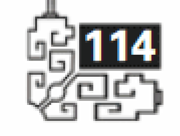


图 7-22　两个 div 元素之间的边距重叠在一起



7.5.4　边距实现对齐功能

margin 属性除了能控制元素的边距外，还有一个非常有用的功能——控制元素水平方向的对齐方式。请看示例：

```
div{
    width:70px;
    height:70px;
    border:2px solid black;
    background-color:yellow;
}
div.left{
    margin:0 auto 0 0;
}
div.center{
    margin:0 auto;
}
div.right{
    margin:0 0 0 auto;
}

<div class="left">左对齐</div>
<div class="center">居中对齐</div>
<div class="right">右对齐</div>
```

.left 的右边距设为 auto 后，元素将居左对齐，.right 的左边距设为 auto 后，元素将居右对齐，.center 中的左右边距都为 auto，元素将居中对齐。效果如图 7-23 所示(如果页面在 IE 中并没有出现期望的结果，请确认是否给页面添加了正确的文档类型声明)。

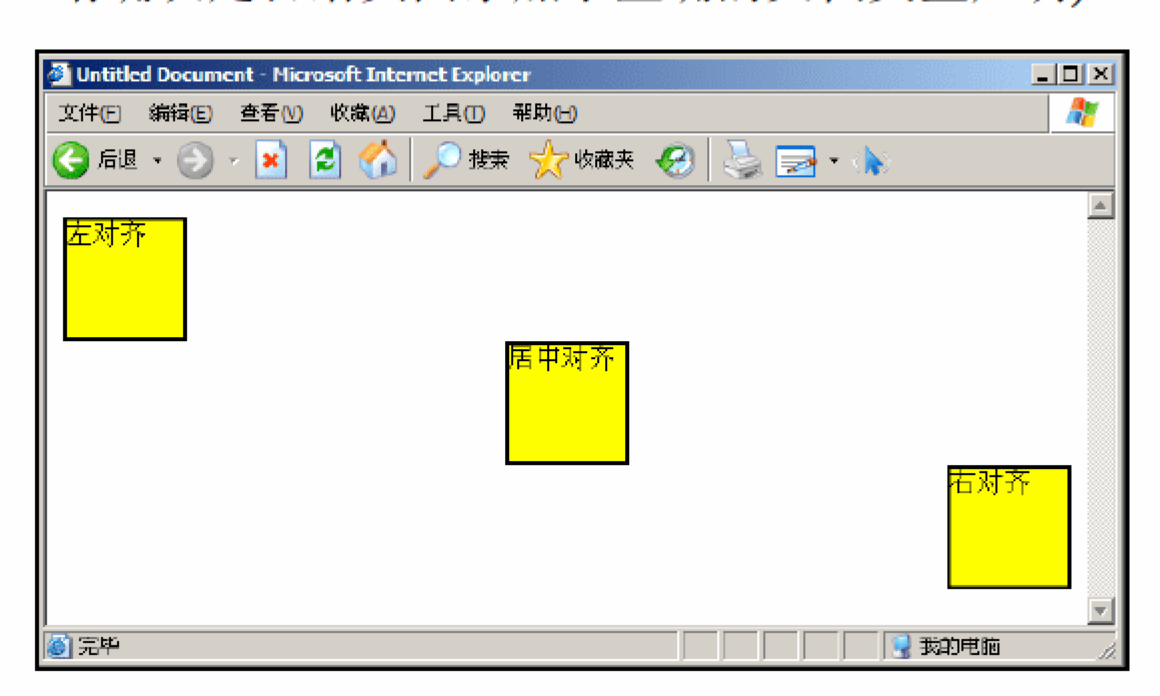
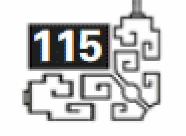


图 7-23　使用 margin 属性控制元素水平对齐方式

7.6　overflow 属性

当我们设定元素内容区域的宽度和高度时，元素实际所包含的内容可能会超出所设定的范围，默认情况下，超出的部分依旧可见。如果想改变它，可使用 CSS 提供的 overflow 属性。



overflow 的属性值可以是 visible、hidden、scroll 和 auto，默认值为 visible。下面的代码展示了这 4 个属性值的不同含义：

```
p{
    width:100px;
    height:100px;
    float:left;
    margin:10px;
    background-color:yellow;
}
p.visible{
    overflow:visible;
}
p.hidden{
    overflow:hidden;
}
p.scroll{
    overflow:scroll;
}
p.auto{
    overflow:auto;
}

<p class="visible">有时,元素的内容区域小于元素实际所包含的内容,使用 CSS 提供的 overflow 属性可控制超出内容区域的部分该如何显示。</p>
<p class="hidden">有时，元素的内容区域小于元素实际所包含的内容，使用 CSS 提供的 overflow 属性可控制超出内容区域的部分该如何显示。</p>
<p class="scroll">有时，元素的内容区域小于元素实际所包含的内容，使用 CSS 提供的 overflow 属性可控制超出内容区域的部分该如何显示。</p>
<p class="auto">有时，元素的内容区域小于元素实际所包含的内容，使用 CSS 提供的 overflow 属性可控制超出内容区域的部分该如何显示。</p>
```

p 元素的长度和宽度都被限制在 100px，但是其中的文字会超出 p 元素所包含的范围，通过改变 overflow 属性可达到不同的效果，如图 7-24 所示。

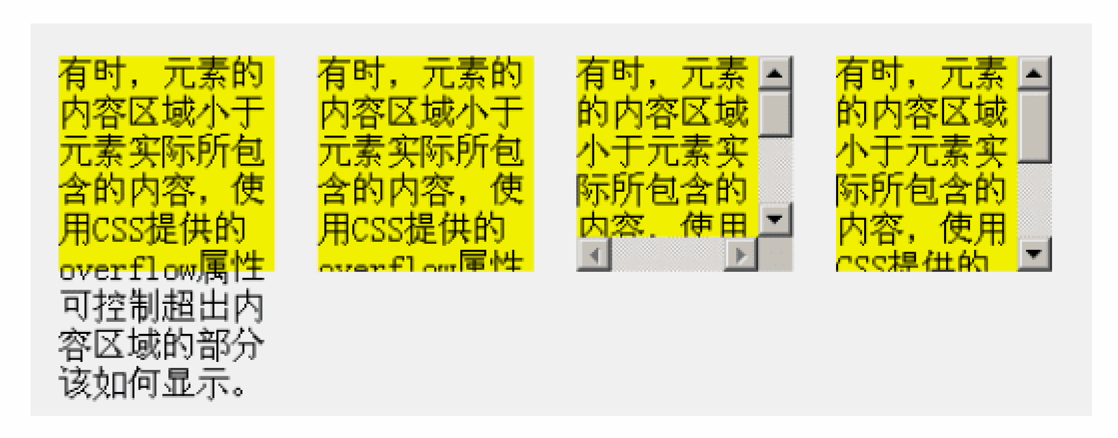


图 7-24　overflow 属性不同属性值的效果(IE7)

overflow 属性值为 visible 时，超出的内容将会显示出来，hidden 将隐藏超出的部分，auto 会在水平和竖直方向添加滚动条，scroll 会在需要的时候添加滚动条。



7.7　元素类型和 display 属性

在第一篇中，我们讲到(X)HTML 元素可分为两大基本类型：块级(Block-level)元素和内联(Inline)元素。像 div、p 一类会独占一行的元素就是块级元素，而 span、a 一类的不产生换行的元素就是内联元素。

盒模型相关的属性(边框、填充和边距属性)对两种类型的元素都是起作用的，但是对于内联元素，个别浏览器的显示效果不尽相同。而 7.4 节介绍的有关维度的属性只适用于块级元素。有没有办法改变不同类型元素的显示方式？CSS 的 display 属性就可以做到。

display 属性的作用就是改变(X)HTML 元素的显示方式，它的属性值相当多，这里只介绍与本章相关的几个：block、inline、inline-block 和 run-in，其余属性值的含义和用法会在后续章节中陆续地介绍。

7.7.1　block 和 inline

使用声明 display:block 和 display:inline 就可以使任何元素满足块级或内联元素的显示特性。请看如下示例：

```
span{
    height:100px;
    width:300px;
    background-color:yellow;
}

<span>span 元素默认为内联元素，height 和 width 属性对其不起作用。</span>
```

此时，代码效果如图 7-25 所示，span 元素仍按照内联元素的方式显示，height 和 width 属性都没有起作用。

span元素默认为内联元素，浏览器会按照内联元素显示方式渲染该元素。另外，height和width属性对span也不起作用。

图 7-25　内联元素的默认显示效果，维度相关的属性不起作用

如果给 span 元素添加 display:block;声明，则浏览器会完全按照块级元素显示属性渲染该元素(见图 7-26)。

span元素默认为内联元素，浏览器会按照内联元素显示方式渲染该元素。另外，height和width属性对span也不起作用。

图 7-26　更改其 display 属性为 block 后的显示效果

同样，更改块级元素的 display 属性为 inline 后，浏览器会依照内联元素进行处理。

7.7.2 inline-block

inline-block 类型的元素含有内联元素和块级元素的部分特性，元素如同文本一样会紧挨着其他元素，前后不产生换行；对于每个单独的元素而言，它们又满足块级元素的特性，height 和 width 属性均能对其产生作用。我们可以把这种元素成为“内联块”型元素。

但遗憾的是，各个浏览器对 inline-block 的支持都不是很好，例如 IE 中只对原本是内联元素起作用。Firefox3 之前版本的浏览器则不识别这个属性值，仍按照原来的方式显示元素。

请看下面的示例：

```
body{
    font-size:12px;
}
div{
    margin-bottom:10px;
    border:1px solid black;
}
p{
    width:140px;
    border:1px dashed gray;
}
div#div2 p{
    display:inline;
}
div#div3 p{
    display:inline-block;
}

<div id="div1">块级元素会独占一行，前后会产生换行。<p>内联元素在其前后不会产生换行，元素会紧挨着其他元素。</p>“内联块”型的元素也不产生换行，但每个单独的元素又满足块级元素的特性。</div>
<div id="div2">块级元素会独占一行，前后会产生换行。<p>内联元素在其前后不会产生换行，元素会紧挨着其他元素。</p>“内联块”型的元素也不产生换行，但每个单独的元素又满足块级元素的特性。</div>
<div id="div3">块级元素会独占一行，前后会产生换行。<p>内联元素在其前后不会产生换行，元素会紧挨着其他元素。</p>“内联块”型的元素也不产生换行，但每个单独的元素又满足块级元素的特性。</div>
```

如图 7-27 所示为以上代码在 Opera 浏览器中的显示效果。

延伸： 在 Firefox 浏览器中可以使用 Mozilla CSS Extensions 提供的-moz-inline-stack 属性达到 inline-block 的显示效果。请参考如下地址：

http://developer.mozilla.org/en/docs/CSS_Reference:Mozilla_Extensions

块级元素会独占一行，前后会产生换行。
内联元素在其前后不会产生换行，元素会紧挨着其他元素。
“内联块”型的元素也不产生换行，但每个单独的元素又满足块级元素的特性。

块级元素会独占一行，前后会产生换行。内联元素在其前后不会产生换行，元素会紧挨着其他元素。“内联块”型的元素也不产生换行，但每个单独的元素又满足块级元素的特性。

块级元素会独占一行，前后会产生换行。内联元素在其前后不会产生换行，元素会紧挨着其他元素。“内联块”型的元素也不产生换行，但每个单独的元素又满足块级元素的特性。

图 7-27　block、inline 和 inline-block 属性值的作用(Opera 浏览器)

7.7.3　run-in

run-in 可以让块级元素以内联方式融入到其后的元素中，比如下面的代码会产生如图 7-28 所示的效果(Opera 浏览器)：

```
p.runin{
    font-family:"Times New Roman", serif;
    display:run-in;
    border:1px dotted gray;
}

<p class="runin">run-in</p><p>类型的元素会以内联方式融入到后面的元素当中。</p>
```

run-in类型的元素会以内联方式融入到后面的元素当中。

图 7-28　run-in 属性值的作用(Opera 浏览器)

延伸： 以下这个网址总结了 display 属性的各个属性值应该达到的显示效果，并给出了当前一些主流浏览器对 display 属性的支持程度：http://www.quirksmode.org/css/display.html

7.8　小　　结

任何(X)HTML 元素都被浏览器视为一个个的矩形区域，在 CSS 中用盒模型(Box Model)这个概念来描述每个元素。盒模型包含如下 4 个部分：内容、填充、边框和边距。

内容区域由元素本身包含的内容多少而定，内容外面是填充，表示内容到边框之间的空间，而边距位于边框外侧，表示本元素与其他元素之间的距离。

width 和 height 属性可以控制盒模型中内容区域的宽度和长度，而整个盒子的宽度和长度需要再加上填充、边框和边距所占的空间。CSS 还提供了设定最小、最大维度的属性。

(X)HTML 元素分为两大基本类型：块级元素和内联元素，不同类型元素的显示方式各异。CSS 的 display 属性可以更改元素的显示方式，不管它们本身是什么类型的元素。

下一章将向读者介绍有关浏览器兼容性问题以及相应的解决方案。



第 8 章　解决浏览器兼容性问题

最让 Web 设计人员感到头疼的就是处理众多浏览器的兼容性，由于不同浏览器对 CSS 支持程度和解析方式的不同，同一页面在不同浏览器中的效果往往相去甚远。本章将向读者介绍一些解决浏览器兼容性问题的基本方法，并说明如何针对特定的浏览器创建样式。

在后面的章节中，我们随时会遇到各种各样的兼容性问题，使用本章介绍的方法就可以进行解决。读者也可以先跳过本章继续阅读后面的内容，当遇到问题时再来参考相关的内容。

本章主要内容

- IE 的条件注释
- 常用的 CSS Hacks 技术
- 如何给特定浏览器添加样式
- 如何正确使用 Hacks 技术

8.1　条 件 注 释

条件注释(Conditional Comments)是 IE 浏览器所特有的，它能够对浏览器的类型和版本进行判定。

8.1.1　条件注释的基本结构

条件注释有以下两种形式(同时也列出了(X)HTML 中一般注释的形式)：

```
<!-- IE 支持的条件注释结构 -->
<!--[if 表达式]> HTML <![endif]-->
<![if 表达式]> HTML <![endif]>
```

第一种条件注释中的内容可以被 IE5 及其以后版本的浏览器支持，当表达式为真时，其中的内容就会被浏览器解析，否则浏览器会忽略掉其中的内容。第二种条件注释中的内容除了能被 IE5 及后续版本解析外，还能被其他浏览器所识别，但是这些浏览器不会在意表达式的真假。

表达式由特征(Feature)、操作符(Operator)和数值(Value)组成，表 8-1 总结了表达式中的合法内容以及它们的用法和含义。

表 8-1　表达式中的合法内容以及它们的含义

项	举　例	说　明
IE	[if IE]	IE 为特征值，表示是 IE 浏览器
value	[if IE 6]	整数或浮点数表示浏览器的版本号，版本号可精确至小数点后面 4 位

续表

项	举　例	说　明
!	[if !IE]	表示“非(NOT)”
lt	[if lt IE 5.5]	表示“低于(less than)”
lte	[if lte IE 6]	表示“低于或等于(less than or equal)”
gt	[if gt IE 5]	表示“高于(greater than)”
gte	[if gte IE 6]	表示“高于或等于(greater than or equal)”
( )	[if !(IE 6)]	表示子表达式
&	[if (gt IE 5) & (lt IE 7)]	表示“与(AND)”
|	[if (IE 6) | (IE 7)]	表示“或(OR)”
true	[if true]	表示“真(TRUE)”
false	[if false]	表示“假(FALSE)”

8.1.2　条件注释举例

1. 判断浏览器类型

请看示例：

```
<!--[if IE]>
<h1>您正在使用微软公司的 Internet Explorer 浏览器</h1>
<![endif]-->
<![if !IE]>
<h1>您正在使用非微软公司的浏览器</h1>
<![endif]>
```

我们分别使用 IE6 和 Firefox 2.0 打开这个页面。在图 8-1 中，上图显示了第一个条件注释中的文字，因为当前浏览器是 IE，条件表达式为真，对于第二个注释，表达式返回为假，因此 IE6 没有显示其中的文字。Firefox 浏览器只能解析第二种写法的条件注释，因此显示了其中的文字。需要注意的是，Firefox 等非 IE 浏览器不能对表达式的返回值进行判断，所以不要从这些浏览器的角度出发添加逻辑代码。

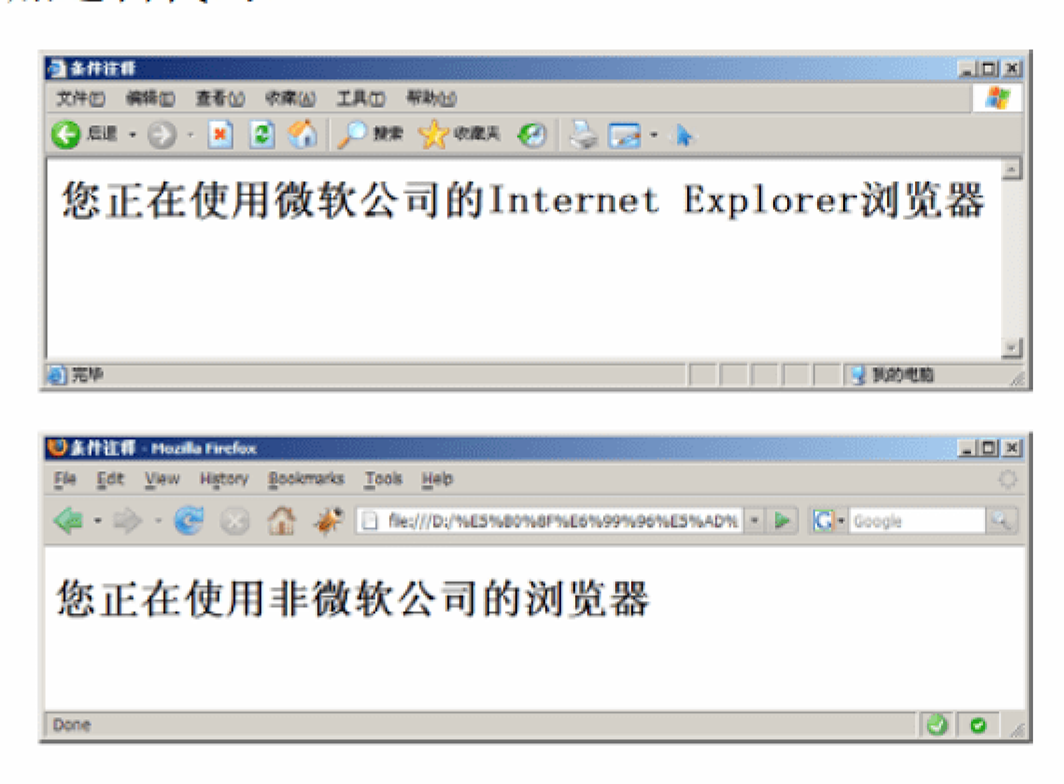
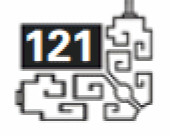


图 8-1　使用条件注释判断浏览器的类别



2. 判断浏览器版本

通过条件注释，我们可以很方便地给 IE 浏览器或不同版本的 IE 浏览器添加特定的样式，请看示例(这里给出完整的代码)：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>条件注释</title>
<style type="text/css">
p{
    padding:10px;
    color:lime;
    background:black;
}
</style>
<!--[if IE 7]>
<style type="text/css">
p{
    color:yellow;
    background:orange;
   font-size:0.8em;
}
</style>
<![endif]-->
<!--[if IE 6]>
<style type="text/css">
p{
    color:white;
    background:red;
   font-size:1.2em;
}
</style>
<![endif]-->
</head>

<body>
    <p>条件注释使得给 IE 浏览器或不同版本的 IE 浏览器添加特定的样式成为可能。</p>
</body>
</html>
```

分别用 Firefox、IE7 和 IE6 打开该页面，从图 8-2 可以看出通过条件注释可实现单独给特定版本的 IE 浏览器指定样式。



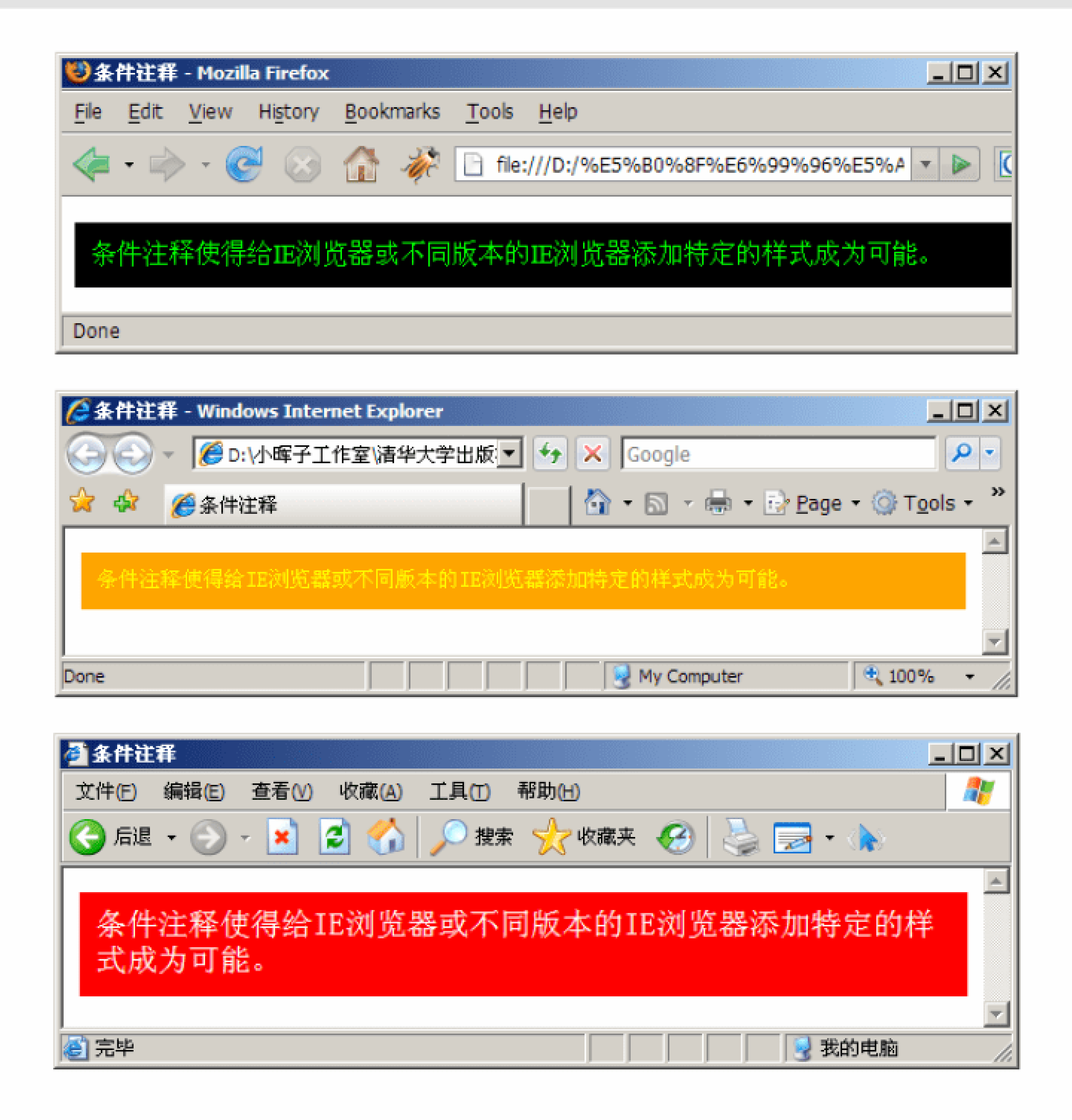


图 8-2　通过条件注释给不同版本的 IE 浏览器指定特定样式(由上至下依次为 Firefox、IE7 和 IE6)

8.2　CSS Hacks

CSS Hacks 是指利用不同浏览器对 CSS 支持程度和解析方式的不同来解决浏览器存在的问题。

8.2.1　利用选择符

IE6、IE7 和 Firefox 对 CSS 选择符的支持程度不同，因此可利用它来给不同浏览器提供专用的 CSS 样式。表 8-2 总结了不同浏览器对选择符的支持情况。

表 8-2　各浏览器对 CSS 选择符的支持程度

选 择 符	IE6	IE7	Firefox2
*(通用选择符)	支持	支持	支持
>(子选择符)	不支持	支持	支持
+(邻接兄弟选择符)	不支持	支持	基本支持
[attr](属性选择符)	不支持	支持	支持

比如我们可以利用属性选择符为 IE6 和 Firefox 添加不同的代码：

```
div.highlighted{
    color:red;           /* IE6 中，颜色为红色 */
}
div[class="highlighted"]{
    color:orange;        /* IE7、Firefox 中，颜色为橙色 */
}
```

由于以上两条 CSS 规则的确定度相同(请参考第 6 章的相关内容)，Firefox 支持后面的属性选择符，因此 IE6 中 div 元素中文字为红色，Firefox 则采用后面的规则显示为橙色。注意只有在选择符的确定度相同时，才能通过规则出现的先后顺序来处理兼容性。比如下面的写法就不能对 IE6 和 Firefox 进行区别处理：

```
div#box1{
    color:red;           /* IE6、IE7 和 Firefox 中，颜色为红色 */
}
div[id="box1"]{
    color:orange;        /* 此声明不起作用，属性选择符确定度小于 id 选择符 */
}
```

由于属性选择符的确定度小于 id 选择符，任何浏览器都不会应用此规则，尽管它位于使用 id 选择符的规则之后。

子选择符也可以使我们能够区分 IE6 与 Firefox，例如：

```
div .normal{
    color:green;         /* IE6 中颜色为绿色 */
}
div>.normal{
    color:blue;          /* IE7 和 Firefox 中颜色为蓝色 */
}
```

8.2.2 利用!important 声明

IE6 对!important 声明支持不够完善，当同一 CSS 规则中存在两条相同的样式声明时，IE6 总会采用后出现的声明(即使前面的样式使用了!important 声明)。比如：

```
p{
    background:red !important;  /* IE7 以及 Firefox 中，p 元素背景色为红色 */
    background:green;           /* IE6 中，p 元素背景色为绿色 */
}
```

8.2.3 利用@import 规则

CSS 不只用于控制 Web 页面如何显示在浏览器中，它还能控制页面输出到其他媒介中的样式。第 17 章将介绍的@media 规则就能指定样式所适用的媒介。当使用@import 规则导入外



部样式表时，可同时指定它所适用的媒介，比如：

```
@import url("all.css") all;         /* 适用于所有媒介 */
@import url("print.css") print;     /* 只适用于打印媒介 */
```

IE 浏览器不支持带有媒介定义的@import 规则，因此可利用这一点为非 IE 浏览器添加特定的代码，比如：

```
/* non-ie.css 中的样式将适用于所有非 IE 浏览器 */
@import url("non-ie.css") all;
```

8.3　正确使用 Hacks 技术

8.3.1　避免乱用 Hacks

CSS Hacks 技术为解决浏览器兼容性问题提供了方便，它是那些富有经验的开发人员多年总结出来的，并不属于 CSS 的标准。有时候，浏览器之间的细微差别是可以接受的，或者由于其他技术的限制，使我们开发的 Web 应用只能运行于某一种浏览器上，这时我们就没有必要使用 Hacks。

另外，有一些 Hacks 技术已经严重影响了代码的正确性，它们利用浏览器解析样式代码中存在的某些问题来实现 Hacks 技术，大部分违反了 CSS 规范，属于非法的属性，因此不建议使用。

8.3.2　良好的习惯

将用于解决浏览器兼容性问题的代码放在独立的文件中是一个良好的习惯，这样做使样式表更具模块化，便于浏览和维护。利用(X)HTML 的 link 元素或者 CSS 的@import 规则可以将这些外部样式导入到所需页面中，比如：

```
<link rel="stylesheet" type="text/css" href="css/ForAll.css" />
<!--[if IE 6]>
<link rel="stylesheet" type="text/css" href="css/ForIE6Only.css" />
<![endif]-->
<!--[if IE 7]>
<link rel="stylesheet" type="text/css" href="css/ForIE7Only.css" />
<![endif]-->
```

所有的浏览器都将应用 ForAll.css 中的样式规则，而后面通过使用条件注释和 link 元素，IE6 和 IE7 将分别使用针对自己的 CSS 文件。

另外，应当在 Hacks 代码中添加适当的注释以说明代码的用途，解决了什么问题，使他人能较快地理解你的代码。



8.4 小　　结

在理想状况下，一份正确的样式代码能在每个支持 CSS 的浏览器中正常工作。但遗憾的是，我们并不是生活在理想世界中，浏览器存在很多问题，而且对 CSS 支持程度和解析方式不同。为了创建能够在各种浏览器上显示相同样式的页面，Web 开发人员通过 Hacks 技术能够选择性地对不同浏览器应用不同地规则。

本章介绍了 IE 浏览器中的条件注释和一些常用的 CSS Hacks 技术。

条件注释可以判定用户是否使用 IE 浏览器和它们的确切版本，通过它可以为 IE 添加特定的样式。

利用选择符的兼容性、IE 对!important 声明和@import 规则支持的不完整性，可添加针对浏览器的特定样式。

下一章将进入本书的第三篇——CSS 样式设计。第 9 章将介绍 CSS 对文本样式的控制。



第三篇　CSS 样式设计

第 9 章　文 字 处 理

尽管现在的网页都含有大量的图片、Flash 动画甚至是影片剪辑，但文字仍然是一种最重要的表达信息的方式。新闻、论文、科技情报、问题解答等信息大部分还是要依赖于文字进行表达。阅读文字信息的舒适程度直接影响浏览者的心理感受，因此优美的文字样式可以让页面显得更专业，可以使访问者方便而舒适地获取所需的信息。

CSS 提供了许多专门用来控制文字样式的属性，比如字体、大小、段落行间距等。这些样式属性分为两大类，一类是针对字体(Font)的，另一类是针对文本(Text)的。本章将介绍所有与文字相关的样式以及如何在文字设计中使用它们。

本章主要内容

- 文字样式在 Web 设计中的重要性
- 如何有效使用字体族名和通用字体族
- 如何使用粗体、斜体等风格样式
- 如何控制段落文字样式

9.1　字体族和字号

本节介绍的样式属性主要用于控制文字最基本的外观，包括字体族、字号和颜色。

9.1.1　ClearType 技术

在讲解字体样式之前，先介绍一项与字体显示相关的技术，这就是微软公司提出的 ClearType。该技术用来提高字体显示的效果，使字体看上去更饱满、更清晰。图 9-1 展示了未开启和开启 ClearType 时文字显示的效果。

不使用ClearType效果

使用ClearType效果

图 9-1　未使用 ClearType 和使用 ClearType 时字体效果的对比

图 9-2 是放大后看到的情形，锯齿化边缘被一些彩色的像素所填充以达到抗锯齿效果。ClearType 只对一部分字体有效，比如图 9-1 中的宋体字就不受影响。



图 9-2　放大后效果对比图(上面为未使用 ClearType，下面使用了 ClearType)

用户系统开启 ClearType 效果后，浏览长篇英文页面时的感受还是会有很大不同的(见图 9-3 和图 9-4)。图 9-3 文字边缘显得比较圆润，图 9-4 的文字明显有锯齿。

W3C Candidate Recommendation

This document is also available in these non-normative formats: plain text, gzip'ed tar file, zip file, gzip'ed PostScript, PDF. See also **translations**.

Copyright © 2006 W3C® (MIT, ERCIM, Keio), All Rights Reserved. W3C liability, trademark and document use rules apply.

Abstract

This specification defines Cascading Style Sheets, level 2 revision 1 (CSS 2.1). CSS 2.1 is a style sheet language that allows authors and users to attach style (e.g., fonts and spacing) to structured documents (e.g., HTML documents and XML applications). By separating the presentation style of documents from the content of documents, CSS 2.1 simplifies Web authoring and site maintenance.

CSS 2.1 builds on CSS2 [CSS2] which builds on CSS1 [CSS1]. It supports media-specific style sheets so that authors may tailor the presentation of their documents to visual browsers, aural devices, printers, braille devices, handheld devices, etc. It also supports content positioning, table layout, features for internationalization and some properties related to user interface.

CSS 2.1 corrects a few errors in CSS2 (the most important being a new definition of the height/width of absolutely positioned elements, more influence for HTML's "style" attribute and a new calculation of the 'clip' property), and adds a few highly requested features which have already been widely implemented. But most of all CSS 2.1 represents a "snapshot" of CSS usage: it consists of all CSS features that are implemented interoperably at the date of publication of the Recommendation.

CSS 2.1 is derived from and is intended to replace CSS2. Some parts of CSS2 are unchanged in CSS 2.1, some parts have been altered, and some parts removed. The removed portions may be used in a future CSS3 specification. Future specs should refer to CSS 2.1 (unless they need features from CSS2 which have been dropped in CSS 2.1, and then they should only reference CSS2 for those features, or preferably reference such feature(s) in the respective CSS3 Module that includes those feature(s)).

Status of this document

图 9-3　开启 ClearType 后的效果

ClearType 效果默认是关闭的，而且只在 Windows XP 及后续操作系统中存在。如果要打开 ClearType 效果，请在桌面空白处点击鼠标右键，选择“属性”命令，打开显示属性的对话框。然后选择“外观”选项卡，单击“效果”按钮，打开效果对话框。将“使用下列方式使屏幕字体的边缘平滑”一项打上对勾，并在下拉框里选择“清晰”，然后确定即可(见图 9-5)。

W3C Candidate Recommendation

This document is also available in these non-normative formats: plain text, gzip'ed tar file, zip file, gzip'ed PostScript, PDF. See also **translations**.

Copyright © 2006 W3C® (MIT, ERCIM, Keio), All Rights Reserved. W3C liability, trademark and document use rules apply.

Abstract

This specification defines Cascading Style Sheets, level 2 revision 1 (CSS 2.1). CSS 2.1 is a style sheet language that allows authors and users to attach style (e.g., fonts and spacing) to structured documents (e.g., HTML documents and XML applications). By separating the presentation style of documents from the content of documents, CSS 2.1 simplifies Web authoring and site maintenance.

CSS 2.1 builds on CSS2 [CSS2] which builds on CSS1 [CSS1]. It supports media-specific style sheets so that authors may tailor the presentation of their documents to visual browsers, aural devices, printers, braille devices, handheld devices, etc. It also supports content positioning, table layout, features for internationalization and some properties related to user interface.

CSS 2.1 corrects a few errors in CSS2 (the most important being a new definition of the height/width of absolutely positioned elements, more influence for HTML's "style" attribute and a new calculation of the 'clip' property), and adds a few highly requested features which have already been widely implemented. But most of all CSS 2.1 represents a "snapshot" of CSS usage: it consists of all CSS features that are implemented interoperably at the date of publication of the Recommendation.

CSS 2.1 is derived from and is intended to replace CSS2. Some parts of CSS2 are unchanged in CSS 2.1, some parts have been altered, and some parts removed. The removed portions may be used in a future CSS3 specification. Future specs should refer to CSS 2.1 (unless they need features from CSS2 which have been dropped in CSS 2.1, and then they should only reference CSS2 for those features, or preferably reference such feature(s) in the respective CSS3 Module that includes those feature(s)).

Status of this document

图 9-4　未开启 ClearType 时的效果

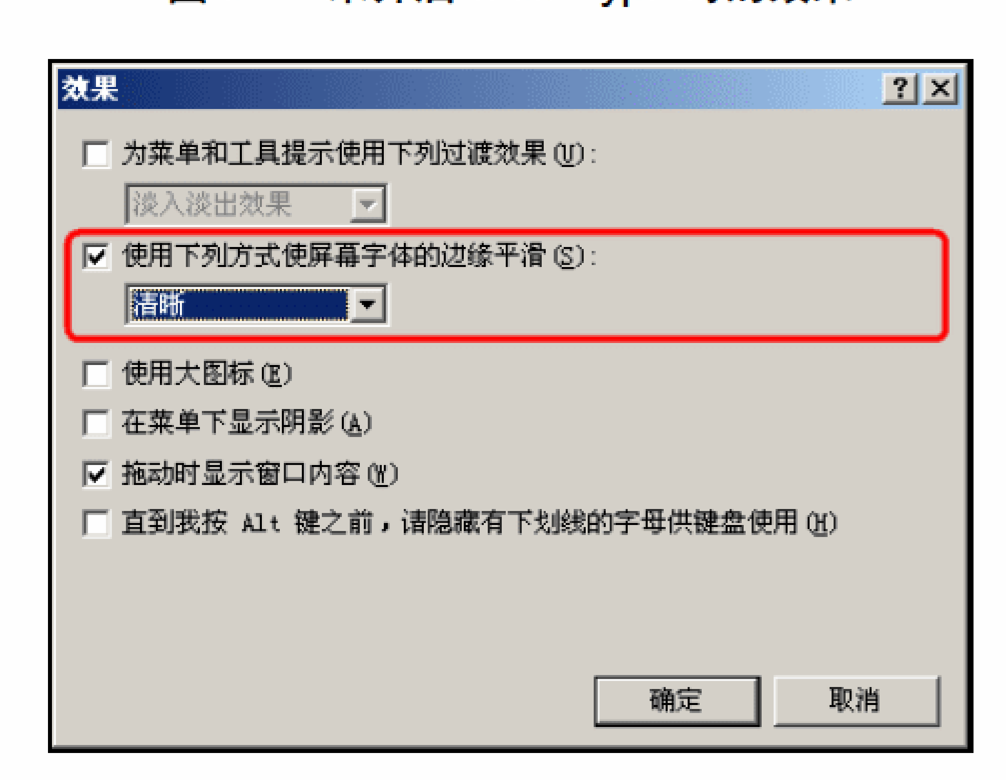
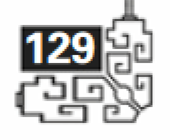


图 9-5　打开 ClearType 功能

微软公司还提供了一个名为 ClearType Tuning 的工具，用来调整 ClearType 细节效果(见图 9-6)。可以用它来打开或关闭 ClearType 功能，调整对比度、LCD 显示条模式等。

Web 设计人员在设计页面时要考虑到用户使用的系统可能没有启动此功能或者甚至没有此功能。由于 Web 页面的兼容性很重要，因此在测试页面显示效果时，要同时考虑到开启和关闭 ClearType 功能的两种情况。



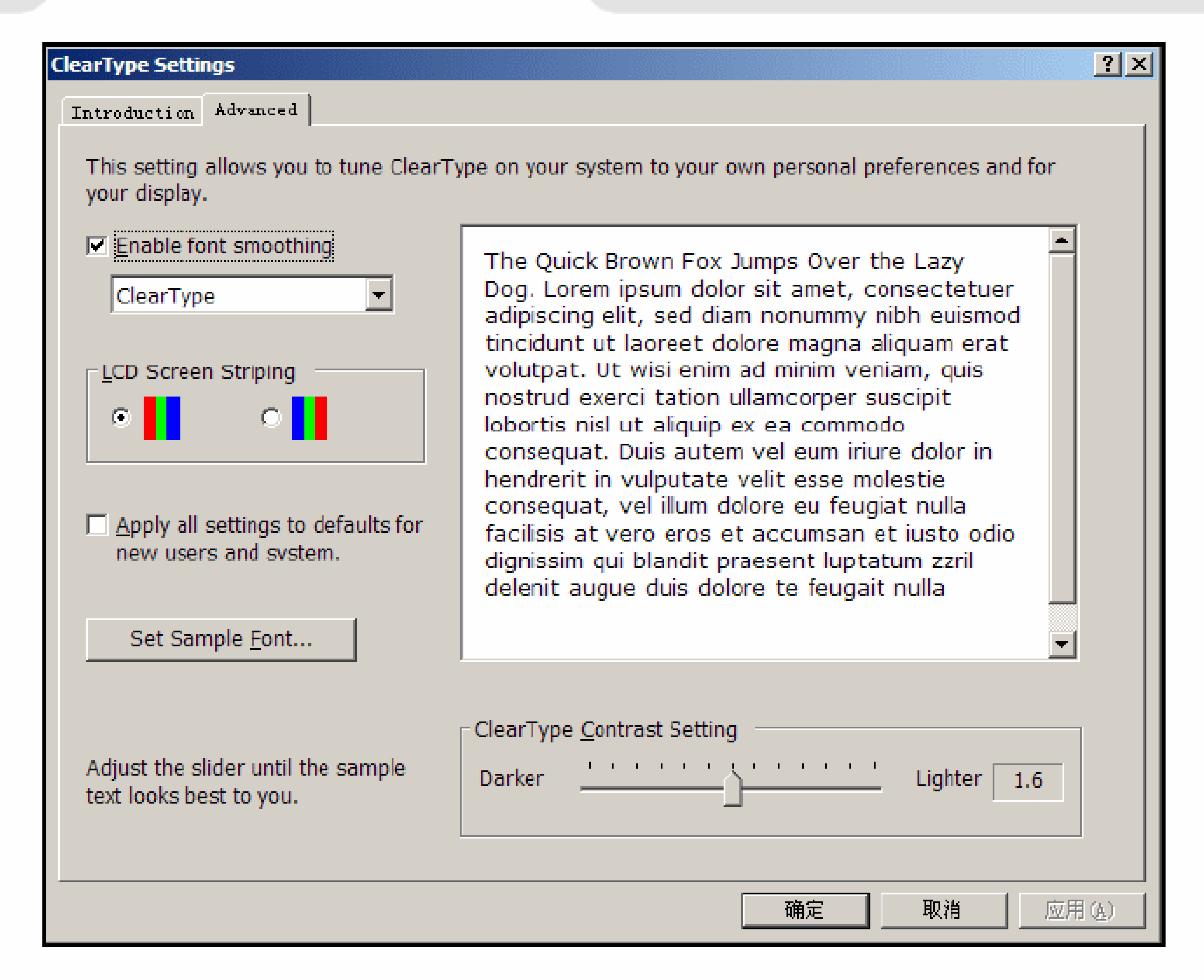


图 9-6　用 ClearType Tuning 工具调整 ClearType 的细节效果

9.1.2　字体族

CSS 使用 font-family 属性来设定字体家族，简单地说就是定义字体外观。常见的中文字体有宋体、黑体、楷体，常见的英文字体有 Times New Roman、Arial、Tahoma 等。font-family 规则的书写格式如下：

```
选择符 {font-family: 字体 1, 字体 2, ...;}
```

浏览器首先查看系统是否有第一个字体，若没有则使用后面的第一个可用字体进行替代。例如：

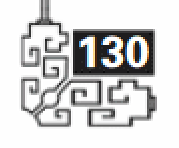
```
p {font-family: "微软雅黑", "黑体", "宋体";}
```

浏览器查找本系统的字体，如果匹配则使用该字体。假如某系统没有微软雅黑字体而有黑体，则浏览器使用黑体进行替代，若黑体也没有就用宋体替代。

字体族的属性值分为两大类，一类是我们先前看到的宋体、Arial 这样的字体族名(family-name)，另一类是通用族(generic-family)。

通过指定确切的字体族名可以控制元素使用哪一种文字样式，前提是这些字体存在于访问者的系统中。图 9-7 列举了一些常见的中英文字体族的外观。

通用字体族是一种容错机制，当系统中没有设计者指定的任何字体时，通用字体族提供了一种可行的替代方案。不同的通用字体族定义了若干特定的字体外观属性，比如要求字体笔划结束时添加装饰线或者要求字符宽度一致等。满足这些特定属性的字体都属于该种通用字体族。设计者可根据需要选择不同的通用字体族进行替代。通用字体族的默认字体在各个操作系统之间可能不尽相同，一些浏览器(如 Firefox)也允许用户自行设置与通用字体族相关的字体(见图 9-8)。



Arial
Times New Roman
Verdana
Georgia
Utopia
Tahoma
Courier New
宋体
黑体
隶书

图 9-7 常见的中英文字体

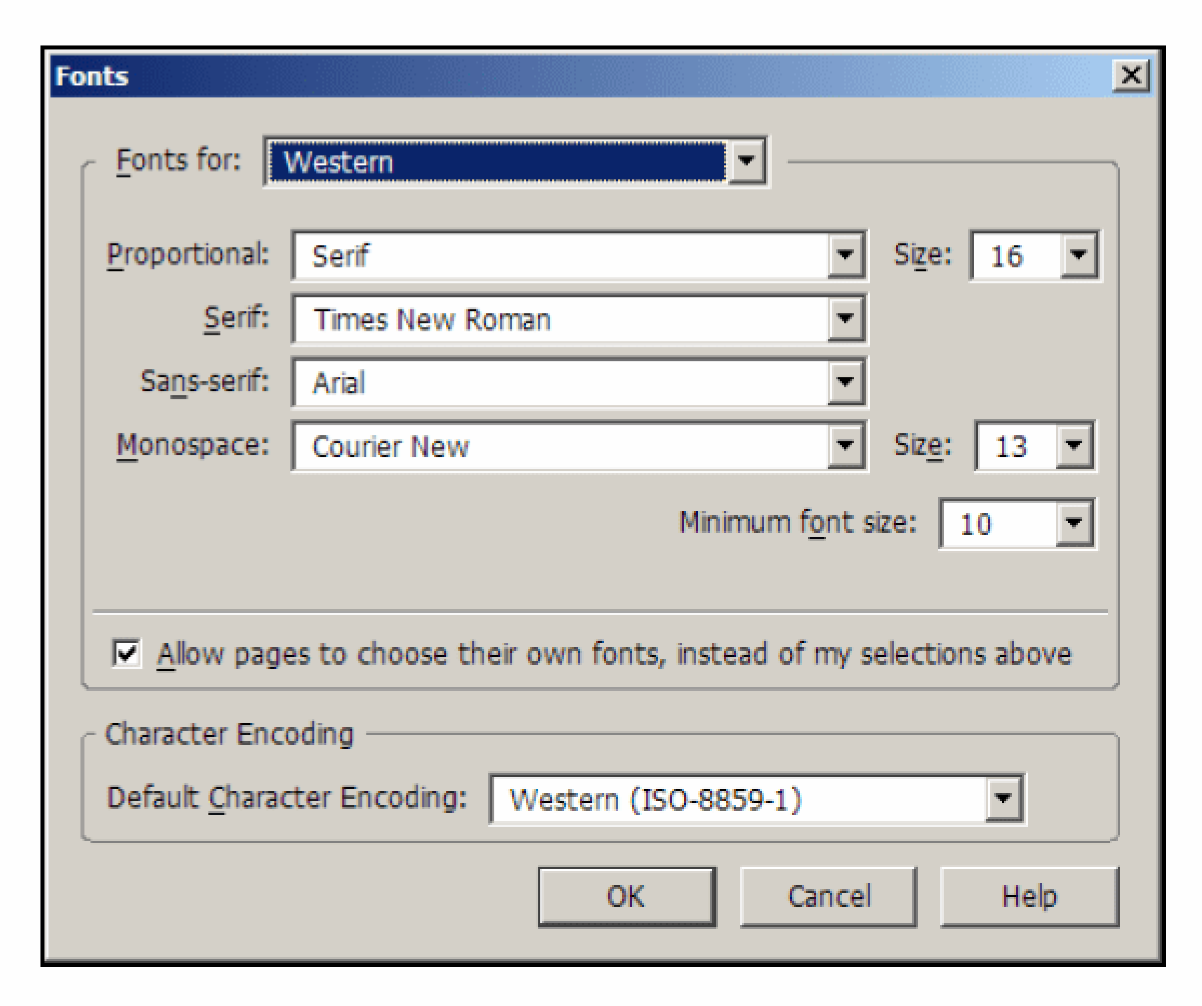


图 9-8 Firefox 中设定通用字体族

CSS 定义了 5 种通用字体族，它们是 serif、sans-serif、cursive、fantasy 和 monospace。每种通用字体族都拥有特定的字体特征，但是具体选用哪一个字体，还要由系统、浏览器或是语言种类确定。请看下面这段代码：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en">
<head>
```

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>字体族</title>
</head>

<body>
<h1 style="font-family:serif;">1. serif family 带衬线字体</h1>
<h1 style="font-family:sans-serif;">2. sans-serif family 无衬线字体</h1>
<h1 style="font-family:cursive;">3. cursive family 草书字体</h1>
<h1 style="font-family:fantasy;">4. fantasy family 装饰字体</h1>
<h1 style="font-family:monospace;">5. monospace family 等宽字体</h1>
</body>
</html>
```

注意，我们在 html 元素中将 lang 属性设置为 en，在 meta 元素中将字符集定为 uft-8。图 9-9 到图 9-11 为以上代码在不同浏览器中显示的效果(由于系统安装的字体不同，读者机器上的显示效果可能与图中所示不符)。

1. serif family 带衬线字体

2. sans-serif family 无衬线字体

3. cursive family 草书字体

4. FANTASY FAMILY 装饰字体

5. monospace family 等宽字体

图 9-9　Internet Explorer 在 Windows XP 中通用字体族的显示

1. serif family 带衬线字体

2. sans-serif family 无衬线字体

3. cursive family 草书字体

4. fantasy family 装饰字体

5. monospace family 等宽字体

图 9-10　Firefox 在 Windows XP 中通用字体族的显示

1. Serif Family 带衬线字体
2. Sans-Serif Family 无衬线字体
3. Cursive Family 草书字体
4. Fantasy Family 装饰字体
5. Monospace Family 等宽字体

图 9-11　Opera 在 Windows XP 中通用字体族的显示

通用字体族所定义的各种属性都是针对西文字符的，但在汉字中也能找出类似的属性。尽管我们的网页大部分内容使用的是中文，但还是免不了包含英文和一些阿拉伯数字，因此需要了解一些有关通用字体族的知识。设计者在定义 font-family 属性时应提供一个通用字体，当用户系统中不存在任何具体的字体时，设计者仍然可以指定一种具有某些样式特征的字体，否则替代字体的选择权就完全由浏览器自己掌握了。

下面举例说明。假如你想将某一段文字字体设为 Times New Roman 这样带有衬线装饰的字体，但是浏览者的系统中没有安装这个字体(尽管这种情况极为少见)，这时你可以在其后添加一个通用字体族 serif，代码如下所示：

```
font-family:"Times New Roman", serif;
```

在学习了下面的内容后你会知道，serif 字体族所包含的字体都是带有衬线的。所以即使没有 Times New Roman，你还能指定一个具有类似样式的字体。

注意：中文字体名和多个单词的字体名要加上双引号。通用字体名是关键字，不能加引号。

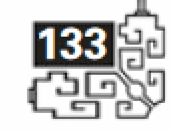
1. serif 字体

serif 在印刷业的术语中含义是衬线，特指加在字母上做装饰的细线，添加细线的字符更容易阅读。与 sans-serif 字体族相比，serif 更能表现出比划粗细的差别，且每个字母宽度是不一样的。比如图 9-12 中字母 T 的第一笔和第二笔的粗细差别就很明显，另外，字母 i 的宽度要小于字母 m 的宽度。

Times New Roman

Georgia 宋体 Song

图 9-12　常见的 serif 字体族字体



serif 字体通常用于正常文本，英文一般用 Times New Roman 作为默认字体，常见的还有宋体、Georgia。注意这些字体的字母在每一笔划末端带有衬线装饰。

2. sans-serif 字体

sans 表示没有，因此 sans-serif 表示不带有衬线的字体，笔划结束处是平的，没有多余的修饰，图 9-13 对比了 serif 与 sans-serif 的一些区别。sans-serif 字体笔划的粗细基本一致，看起来比较匀称。

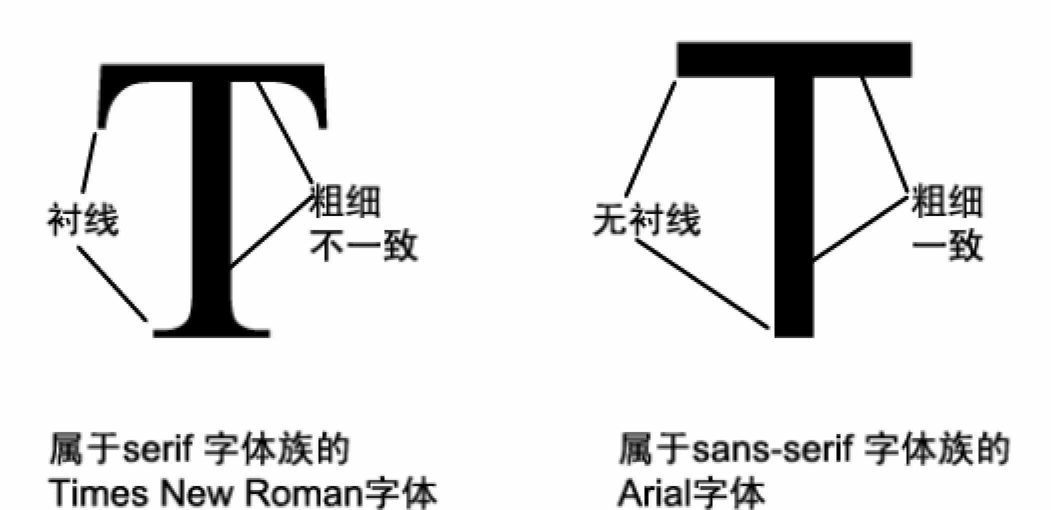


图 9-13 serif 与 sans-serif 字体族的区别

常见的 sans-serif 字体有 Arial、Verdana、Futura、Tahoma、Helvetica 和中文的黑体，如图 9-14 所示。

Arial **Arial Black**

Verdana Futura

Tahoma 黑体

图 9-14 常见的 sans-serif 字体族字体

3. cursive 字体

cursive 在印刷术语中的含义是草体，即一种模仿手写体的印刷字体。该种字体含有连写成分，和用笔快速写出来的效果类似(见图 9-15)。cursive 字体族较难辨认，基本上很少使用，通常一些阿拉伯国家的文字会使用这类字体。

Comic Sans MS

Monotype Corsiva

图 9-15 常见的 cursive 字体族字体

cursive 默认的字体族没有标准，不同系统、不同浏览器之间也存在差异。通常使用的字体有：Comic Sans、Monotype Corsiva、Adobe Poetica 和 Caflisch Script 等。

4. fantasy 字体

fantasy 字体是指那些带有装饰性内容的字体，通常是一些没有规则的、看起来比较奇异的字体，图 9-16 列举的 Critter 和 Studz 字体就属于 fantasy 字体。Fantasy 字体样式的不规则性使其很难进行设计，因此应避免使用。



图 9-16　fantasy 字体

5. monospace 字体

monospace 的含义是等间距，即每个字符之间的距离是相等的，不论是字母 i 还是字母 w，它们所占空间是相等的，很像老式打字机的效果(见图 9-17)。本书所有代码示例使用的字体就是等宽的，这种字体看起来很规整，行与行之间字符是对齐的。

Courier Courier New

Consolas　宋体 Song

图 9-17　常见的 monospace 字体族字体

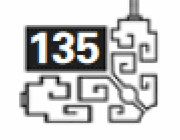
常用的 monospace 字体族是 Courier New，其他还包括 Consolas、宋体、Everson Mono 等。

9.1.3　文字度量

本节将讲解一些有关文字度量的知识，这会使你更好地理解和掌握如何控制文字的大小。该部分内容只适用于西文字符，你也可以略过此部分内容直接阅读后面内容，这只是作为下节内容的补充。

在印刷领域里，文字大小是通过计算字的外形的点(point，简写为 pt)值来衡量的，这种方法称为点制(Point System)。点制有以下三种：欧洲大陆点制，又称为迪多点制(Didot)，1 点相当于 0.376065mm；英美点制，又称十二点制(Pica)，1 点相当于 0.351461mm；此外还有一种英美点制，1 点相当于 1/72 英寸，即 0.352778mm。CSS 使用的就是最后一种。

为了理解如何用点来衡量文字的大小，需要知道如下一些概念：x-高度(x-height)、字母上部(Ascender)、字母下部(Descender)和基线(Baseline)。通过图 9-18 就可以很清楚地看到各个概念所代表的含义。



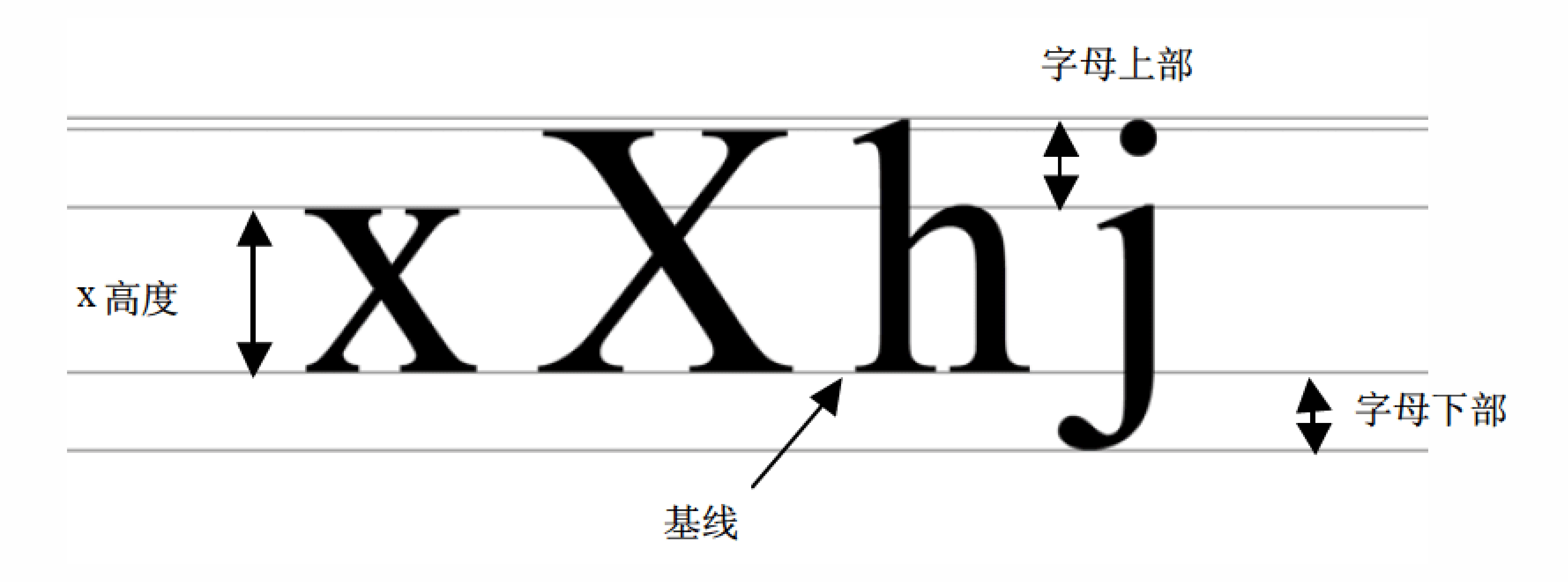


图 9-18　x-高度、字母上部、字母下部与基线

x-高度是指小写字母 x 的竖直方向上的大小，字母上部是指小写字母中高出其他字母的部分，比如 h、b、d 的上部。字母下部是指小写字母比其他小写字母位置低的部分，比如 j、g 的下部。小写字母放置在一条基线上。

注意不同字体的 x 高度可能会不一致，即使它们的点值是相同的。比如 Arial 字体的 x 高度值就要比同点值的 Times New Roman 字体的 x 高度值大。

9.1.4　字体大小

适当地将字体大小相异的文字放置在一起会突出层次感，增强视觉效果。通常标题文字要大于正文文字，表示重要信息的文字大于次要信息的文字。CSS 通过 font-size 属性为文字设定大小，该属性的属性值可以是绝对大小(Absolute Size)、相对大小(Relative Size)、长度值(Length)和百分数(Percentage)。

1. 绝对大小

CSS 中使用如下 7 个关键字来定义文字的绝对大小：xx-small、x-small、small、medium、large、x-large、xx-large。文字具体的大小是由浏览器决定的，medium 所表示的就是浏览器默认的文字大小值，其余属性值都以固定的比例因数放大或缩小这个默认值。CSS1 规范中定义这个比例因数为 1.5，经验证明这个比例过大，因此 CSS2 规定这个值为 1.2。假如 medium 大小为 10pt，则 large 大小就为 12pt 左右。但这也不是绝对的，为了获得更好的阅读效果，比例因数在不同的大小等级会发生变化。

下面这段代码在 IE、Firefox 和 Opera 浏览器中显示的效果如图 9-19 所示：

```
<p style="font-size:xx-small;">xx-small 中文</p>
<p style="font-size:x-small;">x-small 中文</p>
<p style="font-size:small;">small 中文</p>
<p style="font-size:medium;">medium 中文</p>
<p style="font-size:large;">large 中文</p>
<p style="font-size:x-large;">x-large 中文</p>
<p style="font-size:xx-large;">xx-large 中文</p>
```

可以看到在不同浏览器中的文字大小是一致的，但是文字的行间距以及字体的显示效果还是有着细微的差别。



xx-small 中文 xx-small 中文 xx-small 中文
x-small 中文 x-small 中文 x-small 中文
small 中文 small 中文 small 中文
medium 中文 medium 中文 medium 中文
large 中文 large 中文 large 中文
x-large 中文 x-large 中文 x-large 中文
xx-large 中文 xx-large 中文 xx-large 中文

图 9-19　从左向右依次是 IE6、Firefox2 和 Opera9 中的显示效果

2. 相对大小

表示相对大小的属性值只有两个：smaller 和 larger。赋予该属性值的元素的文字大小将相对于其父元素的进行缩放，缩放的比例因数约为 1.2，这与表示绝对大小的属性值之间的比例因数相同。请看下面的示例：

```
p#para1 {font-size:small;}
p#para1 em {font-size:larger;}
p#para2 {font-size:large;}
p#para2 em {font-size:smaller;}
p#para3 {font-size:xx-large;}
p#para3 em {font-size:larger;}

<p id="para1">这段文字包含了一些<em>非常重要</em>的内容</p>
<p id="para2">这段文字包含了一些<em>非常重要</em>的内容</p>
<p id="para3">这段文字包含了一些<em>非常重要</em>的内容</p>
```

我们将三个段落的文字大小分别设置成为 small、large 和 xx-large、其内部的 em 元素文字大小分别设为 larger、smaller 和 larger，由于 larger 和 smaller 设定的缩放比例因数一样，所以前两个段落中 em 元素文字大小是一致的(见图 9-20)。

这段文字包含了一些*非常重要*的内容

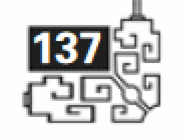
这段文字包含了一些*非常重要*的内容

这段文字包含了一些*非常重要*的内容

图 9-20　设定文字相对大小

3. 长度值

前面介绍过，在传统印刷领域里用点制来描述字符大小，但在 Web 设计中还可以使用其他一些值来描述。CSS 可以用长度值来描述字体大小，我们在第二篇第 5 章中讲解 CSS 度量的时候已经有所介绍，现在再来温习一遍。



长度值带有单位，单位类型可分为以下两种。

- 绝对长度单位：包括 pt(点)、mm(毫米)、cm(厘米)、in(英寸)和 pc(12 点)。
- 相对长度单位：包括 ex(相应字体的 x 高度，即 x-height)、em(相应字体的“font-size”属性值)和 px(像素)。

当使用绝对长度时，绝对不要认为这就是字符显示在屏幕上的大小。绝对长度单位在打印时或在屏幕显示设备的物理尺寸已知时才比较有用。比如 Microsoft Word 就使用点来控制字体大小，所以尽管同样的文档在不同分辨率或者不同大小显示器上显示的大小不尽相同，但是打印出来都是一致的。

相对长度单位中的 ex 表示英文字符 x 的高度，会根据字体族的不同而变化。em 用在 font-size 属性上表示相对其父元素 font-size 的属性值，若父元素尚未指定这个属性，则相对于浏览器默认的文字大小。单位 px 对于我们来说再熟悉不过了，显示器上的图像就是由一个个像素组成的，用 px 作为文字大小单位是 Web 页面设计中最常见的方式之一。

4. 百分数

百分比和 em 单位效果一样，它们都是基于父元素的文字大小来确定本元素文字大小的。100%相当于 1em，150%相当于 1.5em。

5. 字体属性值的选用

目前 Web 页面中的文字设计有两大方式，一种是所有文字大小使用像素作为单位，页面其他元素大部分也使用像素作为单位；另一种就是使用绝对大小、相对大小和长度值中的相对单位进行设计。后一种方式设计的页面可以在 IE6 中实现文字的缩放，而前者不可以。例如雅虎英文页面在 IE 中就是以 small 作为基准大小，其他元素的属性值单位采用百分比或 em，它们都会相对于这个基准值进行变化。图 9-21 展示的是正常字体大小时的页面，当我们在浏览器中设置文字大小为“较大”时，不仅文字放大了，页面的显示区域也拓宽了(见图 9-22)。若查看该页面的 CSS 样式，可发现控制页面宽度元素的 width 属性值为 71.3em，是文字大小的相对值。



图 9-21　正常文字大小的雅虎英文页





图 9-22　将文字大小设置为“较大”后，文字放大，页面变宽

9.1.5　颜色

使用 CSS 给文字添加颜色相当简单，通过 color 属性即可，颜色值可以是 CSS 的关键字和 RGB 值。比如以下两条规则分别将两个段落的文字设为红色和白色。

```
p#p1{color:red;}
p#p2{color:#FFF; background-color:black;}
```

9.2　基本文字样式

除了前面介绍的设定字体族和字体大小外，CSS 还为文字样式提供了如下几个属性：font-style、font-weight、font-variant、text-transform 和 text-decoration。另外，CSS 提供了 font 属性，用来将若干文字属性合并成一条声明。

9.2.1　斜体和粗体

页面中的一些重要文字通常会以不同于其他文本的样式呈现，比如(X)HTML 中 em 元素默认就把文字设置为斜体，h1 一类的标题元素会把文字设置为粗体，现在就来介绍怎样设置斜体和粗体。

1. font-style

font-style 属性有三个可选的值：normal、italic 和 oblique。italic 和 oblique 用来使文字变成

斜体。二者的效果对于大部分字体来说是一样的，某些字体显示略有不同。oblique 属性真正地将文字进行倾斜处理，而 italic 属性使用的是字体中原本包含的具有倾斜效果的字体。但绝大多数开发者使用 italic 属性设置文字倾斜，而 oblique 几乎无人问津。

作者认为斜体效果是字母形式的文字的传统，中文是方块字，不适合采用斜体效果，如果应用斜体反而会增加阅读的困难程度，因此不建议使用。

2. font-weight

font-weight 属性决定了当前字体的粗细程度，其值有如下几种：

- 从 100 到 900，表示文字粗细程度。
- normal 相当于 400。
- bold 相当于 700。
- lighter 和 bolder，分别表示比父元素细或粗一个等级。

实际开发中一般只用 normal 和 bold，即 400 和 700 这两个数值，其他数值的效果与它们是相同的。你可以使用 bold 将默认不加粗的元素中文字加粗或者使用 normal 将默认加粗的元素中文字设为正常。请看如下示例。

(X)HTML 和 CSS 代码：

```
<h1>标题文字默认加粗</h1>
<p>这段文字默认不加粗。</p>
<h1 style="font-weight:normal;">标题文字默认加粗，现在设置为正常效果。</h1>
<p style="font-weight:bold;">这段文字默认不加粗，现在设置为加粗效果。</p>
```

如图 9-23 所示为最终效果。

标题文字默认加粗

这段文字默认不加粗。

标题文字默认加粗，现在设置为正常效果。

这段文字默认不加粗，现在设置为加粗效果。

图 9-23　设置文字为加粗或正常

9.2.2　大小写

西文字符有大小写之分，CSS 也提供了相应的属性：font-variant 和 text-transform。

1. font-variant

该属性是针对西文字符设计的，能让字符全部以大写或小写形式显示，同时还能从字形大小方面对原始文本的大小写形式进行区分。属性值可为 normal 和 small-caps。我们还是通过示例来认识这个属性。

代码如下：



```
body p{
    font-family:"Times New Roman", Times, serif;
    font-variant:small-caps;
}

<p>This is a DEMO for font-variant property.</p>
```

从图 9-24 可看出原本小写的字符全部变成大写了，但是它们明显比原始大写字符小一点，即产生了“小型号”大写字母的效果。

THIS IS A DEMO FOR FONT-VARIANT PROPERTY.

图 9-24　使用 font-variant 产生小型号大写字母效果

注意： 某些字体不支持 small-caps 这种变体形式，这时浏览器会试图通过缩小该字体大写形式来进行显示，但该方法有时会产生异常。

2. text-transform

text-transform 能够强制改变西文字符的大小写形式。它有如下 4 种取值：capitalize、uppercase、lowercase 和 none。capitalize 使每个单词的首字母大写，uppercase 使单词中所有字母均大写，lowercase 使单词中所有字母均小写。none 是默认值，对字符没有任何影响。

请看如下代码：

```
p{font-family:"Times New Roman";}

<p style="text-transform:none;">This is a Demo for text-transform.</p>
<p style="text-transform:capitalize;">This is a demo for text-transform.</p>
<p style="text-transform:lowercase;">This is a demo for text-transform.</p>
<p style="text-transform:uppercase;">This is a demo for text-transform.</p>
```

从图 9-25 看出 none 对原始文本没有影响，下面依次把文字转换为首字母大写、所有字母小写和所有字母大写的样式。

This is a Demo for text-transform.

This Is A Demo For Text-Transform.

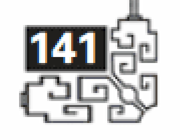
this is a demo for text-transform.

THIS IS A DEMO FOR TEXT-TRANSFORM.

图 9-25　应用 text-transform 转换文字样式

9.2.3　文字装饰

该属性可给文本添加上划线、下划线、删除线和闪烁效果，对应的属性值为 overline、underline、line-through 和 blink。



如图 9-26 所示为 amazon.com 显示图书信息页面的一部分，可以看到链接文字有下划线做装饰，图书原始价格则用删除线进行装饰。

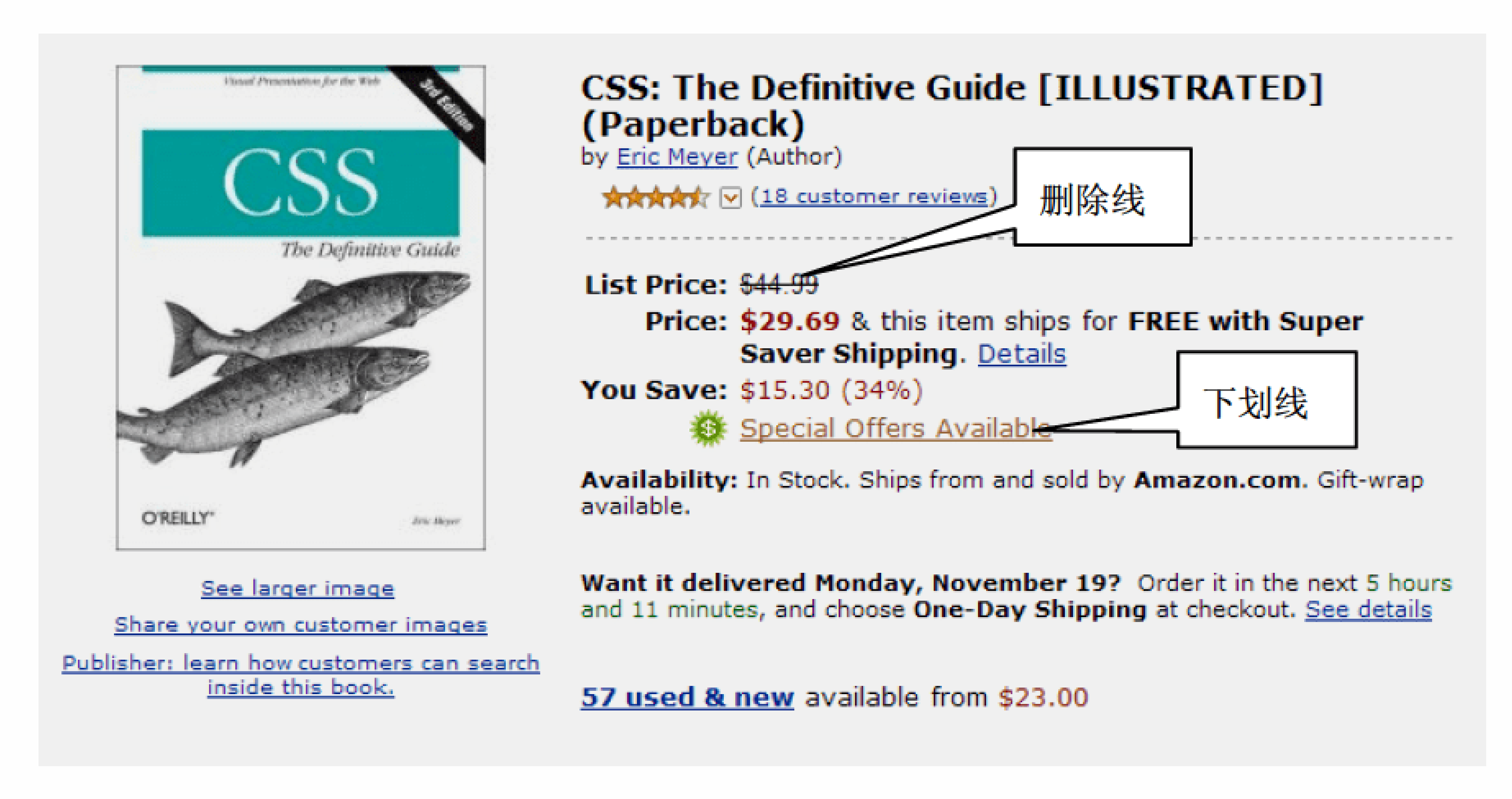


图 9-26　amazon.com 页面使用了 underline 和 line-through 属性

我们还可以将多个属性值写在一条规则中，例如：

```
text-decoration:line-through underline overline;
```

不是所有的浏览器都支持 blink 属性，IE 就不会有闪烁的效果出现。

9.2.4　font 属性

font 属性有两个作用：第一，它简化了字体相关样式的书写；第二，它允许 Web 页面使用和系统设置相关的字体。

1. font 简化字体相关样式的书写

到这里读者已经学习了所有关于字体样式的属性了。如果页面运用了较多的样式，代码看起来比较繁琐，CSS 提供了 font 属性，作为一种缩略形式，它将多个样式的属性值写在一起，用空格相隔。font 属性可包含如下声明：font-style、font-variant、font-weight、font-size、line-height 和 font-family。

属性出现要按照一定的顺序，否则不会影响所控制的元素。首先出现的应是 font-style、font-variant 和 font-weight 属性(这三个属性值可互换位置)，之后是 font-size 和 line-height，最后是 font-family，每个属性不必都出现。由于 font-size 和 line-height 的属性值类型相同，因此采用如下方式进行区分——如果只有一个该类型值出现，则该值表示 font-size，若二者都出现，则采用如下方式书写：font-size/line-height，二者用斜杠相隔。

请看如下示例：

```
p#p1{
    font:oblique 16px "Times New Roman", "宋体", Times, serif;
```



```
}
p#p2{
    font:bold italic 1.5em/1.5 "微软雅黑", sans-serif;
}
p#p3{
    font:small-caps 1.2em/30px Georgia, "Times New Roman", Times, serif;
}

<p id="p1">CSS 文字样式属性众多，使用 font 属性可以将一部分属性集中在一起，使代码更简洁。
</p>
<p id="p2">CSS 文字样式属性众多，使用 font 属性可以将一部分属性集中在一起，使代码更简洁。
</p>
<p id="p3">The font property is the shorthand property for all the other font
properties.</p>
```

如图 9-27 所示为以上代码的效果。

CSS文字样式属性众多，使用font属性可以将一部分属性集中在一起，使代码更简洁。

CSS文字样式属性众多，使用font属性可以将一部分属性集中在一起，使代码更简洁。

THE FONT PROPERTY IS THE SHORTHAND PROPERTY FOR ALL THE OTHER FONT PROPERTIES.

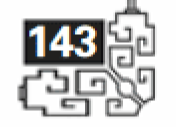
图 9-27　使用 font 属性将各个字体相关属性组合在一起

2. 使用系统字体

font 属性除了可以包含字体相关样式属性外，还能让页面使用系统中设定的字体样式，尤其是你要设计一个嵌入到程序当中的 Web 页面时，使用系统字体可以让页面效果与系统环境更协调一致。font 表示系统字体的属性值和含义，如表 9-1 所示。

表 9-1　font 表示系统字体属性值的含义

font 属性值	含　义
caption	含有标题属性的控件上的字体样式，比如按钮
icon	图标所使用的字体样式
menu	菜单所使用的字体样式
message-box	对话框所使用的字体样式
small-caption	标签控件的字体样式
status-bar	窗口状态栏的字体样式



9.3 段落文字样式

前面介绍的样式都是针对单个字符的，而网站中文字内容最终还是以段落的形式呈现给访问者。CSS 提供了丰富的样式，可灵活设计页面中的段落。

9.3.1 字词间距

CSS 提供了两个用来控制字和词间距的属性：letter-spacing 和 word-spacing。由于中文段落没有词的区分，所以只有 letter-spacing 会对中文段落产生效果。

letter-spacing 属性用来更改单个字符之间的距离，对中文也有效。word-spacing 用来在原有基础上更改单词之间的距离。这两个属性可使用长度值、百分数等任何用于控制文字大小的值进行赋值，前面说过长度值可正可负，负长度值就会在这里派上用场。

请看下面的示例：

```
p#p2{
    letter-spacing:0.4em;
}
p#p3{
    word-spacing:0.4em;
}
p#p5{
    letter-spacing:-1px;
    word-spacing:1.2em;
}

<p id="p1">中文段落没有对词进行区分，因此只有 letter-spacing 样式有效。</p>
<p id="p2">中文段落没有对词进行区分，因此只有 letter-spacing 样式有效。</p>
<p id="p3">中文段落没有对词进行区分，因此只有 letter-spacing 样式有效。</p>
<p id="p4">A paragraph written in English can use both of them.</p>
<p id="p5">A paragraph written in English can use both of them.</p>
```

图 9-28 为最终效果。

中文段落没有对词进行区分，因此只有letter-spacing样式有效。

中 文 段 落 没 有 对 词 进 行 区 分 ， 因 此 只 有 l e t t e r - s p a c i n g 样 式 有 效 。

中文段落没有对词进行区分，因此只有letter-spacing样式有效。

A paragraph written in English can use both of them.

A   paragraph   written   in   English   can   use   both   of   them.

图 9-28　使用 letter-spacing 和 word-spacing

9.3.2　行高

适当地增加段落中文字之间的行距会提高文字阅读的舒适程度，新浪和网易的新闻正文的行距就被设定为 23px，比默认的间距要大，提高了文字的可读性。通过 line-height 属性即可改变行间距的大小。

图 9-29 展示了设定不同行间距所产生不同效果。

行高1.6em：对于Web页面中的大段文字内容，增加行与行之间的距离，设定适当的行间距可提高文字的可读性。CSS提供的line-height属性可用来更改默认的行间距，属性可以是百分比、长度值和数值。

行高300%：对于Web页面中的大段文字内容，增加行与行之间的距离，设定适当的行间距可提高文字的可读性。CSS提供的line-height属性可用来更改默认的行间距，属性可以是百分比、长度值和数值。

行高12px：对于Web页面中的大段文字内容，增加行与行之间的距离，设定适当的行间距可提高文字的可读性。CSS提供的line-height属性可用来更改默认的行间距，属性可以是百分比、长度值和数值。

行高2：对于Web页面中的大段文字内容，增加行与行之间的距离，设定适当的行间距可提高文字的可读性。CSS提供的line-height属性可用来更改默认的行间距，属性可以是百分比、长度值和数值。

图 9-29　line-height 属性可更改行间距

浏览器默认的行间距大约为 120%，因此如果想要增加或减少行间距都要以 120%为基准。百分比和 em 值会根据文字大小调整行间距，即它们之间的比例不会变，如果用 px 作为单位，间距始终是固定的，不会因文字大小变化而变化。

line-height 属性还可以赋给不包含单位的数值，一般情况下数值 1.5 相当于 1.5em 或 150%，但是当 line-height 属性继承给子元素时，数值和其他类型就有区别了。line-height 属性在默认情况下是继承于父元素的，但是继承下来的是固定的计算值。比如父元素字体大小是 14px，line-height 属性为 1.5em，则继承下来的值不是 1.5em，而是 1.5×14px=21px。因此不论子元素的字号为多少，行间距总固定在 21px。请看示例：

```
body{
    font-family:"Times New Roman","宋体";
    font-size:14px;
    line-height:1.5em;
}
p#p2{
    font-size:30px;
}
p#p3{
    font-size:12px;
}
```



```
<p id="p1">line-height 属性在默认情况下是继承于父元素的，但是继承下来的是固定的计算值。比如父元素字体大小是 14px，line-height 属性为 1.5em，则继承下来的值不是 1.5em，而是 1.5 x 14px = 21px。</p>
<p id="p2">文字大小设为 30px：line-height 属性在默认情况下是继承于父元素的，但是继承下来的是固定的计算值。比如父元素字体大小是 14px，line-height 属性为 1.5em，则继承下来的值不是 1.5em，而是 1.5 x 14px = 21px。</p>
<p id="p3" >文字大小设为 12px：line-height 属性在默认情况下是继承于父元素的，但是继承下来的是固定的计算值。比如父元素字体大小是 14px，line-height 属性为 1.5em，则继承下来的值不是 1.5em，而是 1.5 x 14px = 21px。</p>
```

效果如图 9-30 所示。

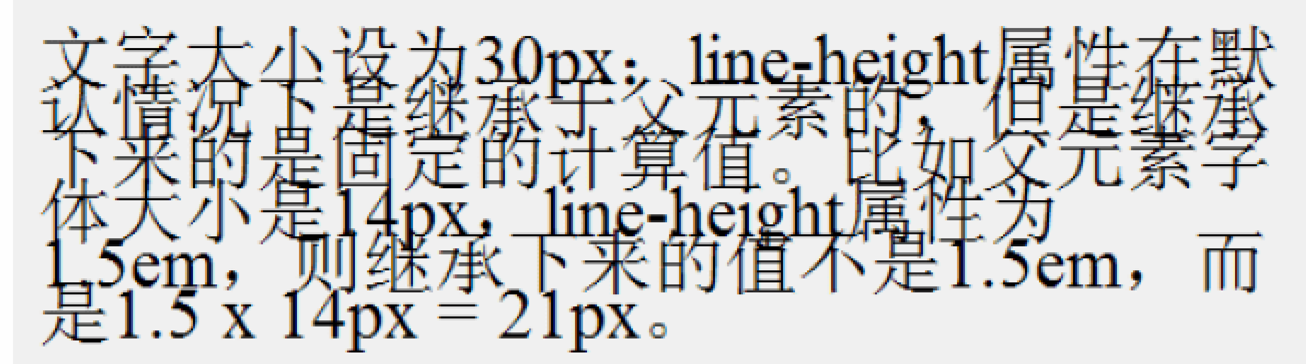


图 9-30　line-height 继承的是计算值

从图 9-30 可看出，不论字体大小如何设定，继承下来的 line-height 属性都是固定的。如果将 body 元素的 line-height 属性改为数值而不是长度值，那么每个元素在继承该属性值时，都要用这个值和本身的字体大小重新进行计算。我们将 body 的样式进行如下更改：

```
body{
    font-family:"Times New Roman","宋体";
    font-size:14px;
    line-height:1.5;
}
```

现在效果如图 9-31 所示。

可以看到子元素的行距都会根据其文字大小进行变化。接下来的示例展示了更改段落的行间距之后，页面效果得到改观。

(X)HTML 代码：

```
<h1>Web 页面中的文字和段落</h1>
<p>文字是向访问者表达信息的重要方式，文字和段落的排版至关重要。对于大段文字内容，我们要选择适合阅读的字体，恰当的颜色，重要信息要用另外一种醒目的样式，与正文区分开，行与行之间设定适当的距离，段首缩进等。访问者即使长时间阅读也不易产生视觉疲劳。</p>
```

我们不添加任何样式，图 9-32 显示了浏览器默认的效果。

line-height属性在默认情况下是继承于父元素的，但是继承下来的是固定的计算值。比如父元素字体大小是14px，line-height属性为1.5em，则继承下来的值不是1.5em，而是1.5 x 14px = 21px。

文字大小设为30px：line-height属性在默认情况下是继承于父元素的，但是继承下来的是固定的计算值。比如父元素字体大小是14px，line-height属性为1.5em，则继承下来的值不是1.5em，而是1.5 x 14px = 21px。

文字大小设为12px：line-height属性在默认情况下是继承于父元素的，但是继承下来的是固定的计算值。比如父元素字体大小是14px，line-height属性为1.5em，则继承下来的值不是1.5em，而是1.5 x 14px = 21px。

图 9-31　line-height 使用数值型

Web页面中的文字和段落

文字是向访问者表达信息的重要方式，文字和段落的排版至关重要。对于大段文字内容，我们要选择适合阅读的字体，恰当的颜色，重要信息要用另外一种醒目的样式，与正文区分开，行与行之间设定适当的距离，段首缩进等。访问者即使长时间阅读也不易产生视觉疲劳。

图 9-32　段落的默认效果

现在我们要调整字体大小和行间距，添加如下样式：

```
body{
    font-family:"Times New Roman", "宋体", Times, serif;
}
h1{
    font-size:18px;
}
p{
    font-size:14px;
    line-height:1.6em;
    color:#343434;
}
```

最终效果如图 9-33 所示。

Web页面中的文字和段落

文字是向访问者表达信息的重要方式，文字和段落的排版至关重要。对于大段文字内容，我们要选择适合阅读的字体，恰当的颜色，重要信息要用另外一种醒目的样式，与正文区分开，行与行之间设定适当的距离，段首缩进等。访问者即使长时间阅读也不易产生视觉疲劳。

图 9-33　添加样式之后的效果



9.3.3 缩进

中文段落习惯于将首行进行缩进处理，一般要缩进两个字符。使用 text-indent 属性就可以实现首行缩进，例如：

```
text-indent:2em;
```

这条规则可将首行缩进两个字符。如果使用 px 作为单位则缩进大小是固定的，使用负值可达到相反的效果，让首行突出来。注意该属性的百分比值和 em 值效果不同，em 值是相对于字体大小的，而百分比是相对于包含文字元素的父元素的宽度。这里举例说明：

```
body{
    font-family:"Times New Roman","宋体";
    font-size:14px;
    line-height:1.5;
}
p{
    margin-left:100px;
    border:1px solid;
}

<p style="text-indent:2em;">中文段落习惯采用缩进进行排版，首行一般会缩进两个字符，使用 text-indent 属性可达到这样效果。</p>
<p style="text-indent:50%;">中文段落习惯采用缩进进行排版，首行一般会缩进两个字符，使用 text-indent 属性可达到这样效果。</p>
<p style="text-indent:-4em;">中文段落习惯采用缩进进行排版，首行一般会缩进两个字符，使用 text-indent 属性可达到这样效果。</p>
```

为了更好地查看效果，我们给 p 元素增加了边框，并设定 100 个像素的左边距，图 9-34 为最终效果。2em 使首行缩进两个字符，50%使首行缩进宽度为 body 元素宽度的一半大小，因为包含 p 元素的父元素就是 body 元素了。-4em 使得首行突出 4 个字符，也就相当于除首行以外的各行缩进了 4 个字符。

中文段落习惯采用缩进进行排版，首行一般会缩进两个字符，使用text-indent属性可达到这样效果。

中文段落习惯采用缩进进行排版，首行一般会缩进两个字符，使用text-indent属性可达到这样效果。

中文段落习惯采用缩进进行排版，首行一般会缩进两个字符，使用text-indent属性可达到这样效果。

图 9-34　text-indent 设定段落首行缩进

如果想对整个段落进行缩进，则可根据实际情况设置盒模型中的边距或填充属性。

9.3.4　对齐方式

1. 水平对齐

页面文字大部分都是左对齐的，一些标题或特殊内容可能会采用居中对齐或右对齐，使用 text-align 可改变文字的对齐方式，可取值为 left、center、right 和 justify。它们的含义如图 9-35 所示。



图 9-35　text-align 的各个属性值的含义

需要说明的是，justify 的含义虽然是两边对齐，但是不影响不满一行的内容(比如最后一行)，由于中文字符都是等宽的，左对齐时不会像图 9-35 所示那样在右侧产生“锯齿”，因此 left 和 justify 对于中文段落来说效果是等同的。

2. 垂直对齐

text-align 处理的是文本的水平对齐，那 CSS 能更改垂直方向上的位置吗？答案是肯定的。CSS 中的 vertical-align 属性尽管不是针对文本设计的，但是可用来控制文本的垂直对齐方式。vertical-align 属性值可以是关键字、百分数和长度值，表 9-2 总结了各属性值及其具体含义。

表 9-2　vertical-align 各属性值的含义

属 性 值	含　义
baseline	本元素基线与父元素基线对齐，若本元素没有基线，则底边与父元素基线对齐
top	本元素顶端与父元素盒模型顶端对齐
middle	本元素垂直方向的中点与父元素的基线加上其 x 高度的一半对齐
bottom	本元素底端与父元素盒模型底端对齐
text-top	本元素顶端与父元素文本顶端对齐
text-bottom	窗口状态栏的字体样式
super	将文本转为上标
sub	将文本转为下标
百分数	使本元素上升(正值)或下降(负值)，移动多少由该百分比乘以 line-height 值决定，“0%”相当于 baseline
长度值	使本元素上升(正值)或下降(负值)，移动多少由该值决定，“0”相当于 baseline

读者可能对 top、bottom、text-top 和 text-bottom 这几个值的含义和区别不是很清楚，请看

下面的示例：

```
p{
    background:yellow;
    vertical-align:baseline;
    font-size:1.4em;
}
span{
    font-size:0.6em;
}
.largeSize{
    font-size:2em;
}
.smallSize{
    font-size:0.5em;
}

<p>这段文字<span class="largeSize">有大</span><span class="smallSize">有小
</span>。现在能区分出<span style="vertical-align:text-top;">text-top</span>、
<span style="vertical-align:top;">top</span>和<span
style="vertical-align:text-bottom;">text-bottom</span>和<span
style="vertical-align:bottom;">bottom</span>的区别了吗？</p>
```

首先，规则将 p 元素背景设为黄色，为了能清晰显示出 p 元素盒子的上下边界，将一些字体设大的目的在于让元素盒子的边界和元素中文本的边界区分得更明显。从图 9-36 很明显地看出含有“text”的属性值都是与文本的边界对齐，否则是与元素边界对齐的。

这段文字有大有小。现在能区分出text-top、top和text-bottom和bottom的区别了吗？

图 9-36　top、text-top、bottom 和 text-bottom 属性

上标或下标在页面中也是比较常见的，比如™、9th、P_n 等。使用 super 和 sub 这两个值就可轻松实现，请看如下示例：

```
span.superScript{
    vertical-align:super;
    font-size:0.7em;
}
span.subScript{
    vertical-align:sub;
    font-size:0.6em;
}

<p><span class="superScript">TM</span>是英文 trademark(商标)的缩写，在国际上使用比较普遍，主要是告诉他人该图形或文字是作为商标使用的，而不是装璜，警示其他生产厂商不要擅自使用。</p>
<p>X<span class="subScript">1</span>+X<span
class="subScript">2</span>+X<span class="subScript">3</span>=?</p>
```

效果如图 9-37 所示。

™是英文trademark（商标）的缩写，在国际上使用比较普遍，主要是告诉他人该图形或文字是作为商标使用的，而不是装璜，警示其他生产厂商不要擅自使用。

$X_1+X_2+X_3=?$

图 9-37　使用 vertical-align 实现上标和下标

在 vertical-align 属性值中，百分数和长度值的含义比较明确，这里就不再举例说明了。

9.3.5　强制换行

有时页面某些内容的宽度需要固定不变，浏览器会在每一行文字的适当位置产生换行，注意，这种换行一定是在单词与单词之间进行的，绝不会在一个单词之内产生换行。但是如果一个单词的长度已经超出了区域所限定的范围，则浏览器不会自动在词内产生换行，于是单词会有一部分位于区域的外侧。例如：

```
p{
    font-family:"Times New Roman", serif;
    border:1px dashed gray;
    width:100px;
    font-size:20px;
}

<p>simple</p>
<p>antidisestablishmentarianism</p>
```

以上代码在 IE 和 Firefox 中效果如图 9-38 所示。在 IE 中，过宽的单词将整个 p 元素的区域“撑”大了，而 Firefox 中 p 元素宽度不变，但是一部分单词超出了所限制的区域范围。

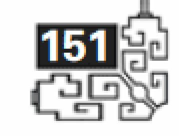
simple	simple
antidisestablishmentarianism	antidisestablishmentarianism

图 9-38　过长的单词破坏了页面效果(左侧为 IE，右侧为 Firefox)

尽管实际中这种情况极少出现，但作为 Web 页面的设计者，应该保证页面在各种极端状况下还能正常显示。对于 IE 浏览器，我们可以使用 word-break 或者 word-wrap 属性，这两个属性都是CSS3中的，目前的Firefox2浏览器还不支持它。将元素的word-break属性设为break-all或者将 word-wrap 属性设为 break-word 即可在 IE 中解决此问题。而对于 Firefox 浏览器则只能暂用 overflow:hidden;声明将超出部分隐藏，尽量保证页面效果。

现在添加如下样式到 p 元素中：

```
p{
    word-wrap:break-word;
    overflow:hidden;
}
```



修改后的效果如图 9-39 所示。

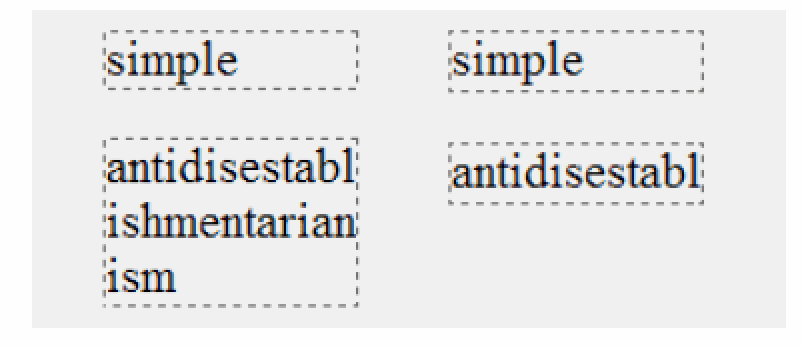


图 9-39　修改后的效果(左侧为 IE，右侧为 Firefox)

9.3.6　其他相关属性

还有一些属性与控制段落样式相关，这些属性包括 white-space、direction、unicode-bidi、text-overflow 和 layout-flow 等。

1. white-space

该属性用来处理元素之内的空白，这些空白可能是一个或多个空格符、换行符或制表符形成的。表 9-3 总结了该属性的取值及其含义。

表 9-3　white-space 各属性值的含义

属 性 值	含　义
normal	按照默认方式处理空白
pre	将所有类型的空白压缩为一个空格，会根据元素大小自动产生换行，这也是浏览器默认的处理方式
nowrap	压缩空白为一个空格，不产生换行，除非遇到 br 元素
pre-wrap	与 pre 效果一致，但会自动产生换行。IE 和 Firefox 不支持
pre-line	压缩空白为一个空格，会自动产生换行。IE 和 Firefox 不支持

IE 和 Firefox 浏览器均不支持 pre-wrap 和 pre-line 属性，因此本节代码示例使用 Opera 浏览器进行展示。请看示例：

```
p{
    font-size:14px;
    font-family:"Times New Roman", "宋体", serif;
    line-height:1.3em;
    width:300px;
    border:1px solid lightgrey;
}

<p style="white-space:normal;">这段  文字拥有很多      空 白，空白可以由     一个
   或多个       空格符、 制   表   符   或 换行符
          产生，white-space 属性可     用来处理 这些空白。
</p>
<p style="white-space:nowrap;">这段  文字拥有很多      空 白，空白可以由     一个
   或多个       空格符、 制   表   符   或 换行符
```



```
        产生，white-space 属性可    用来<br />处理 这些空白。
</p>
<p style="white-space:pre;">这段  文字拥有很多     空 白，空白可以由    一个
  或多个      空格符、  制   表   符   或 换行符
        产生，white-space 属性可    用来处理 这些空白。
</p>
<p style="white-space:pre-wrap;">这段  文字拥有很多     空 白，空白可以由    一个
  或多个      空格符、  制   表   符   或 换行符
        产生，white-space 属性可    用来处理 这些空白。
</p>
<p style="white-space:pre-line;">这段  文字拥有很多     空 白，空白可以由    一个
  或多个      空格符、  制   表   符   或 换行符
        产生，white-space 属性可    用来处理 这些空白。
</p>
```

图 9-40 为 Opera 浏览器中的显示效果。

```
这段 文字拥有很多 空 白，空白可以由 一个 或多
个 空格符、 制 表 符 或 换行符 产生，white-
space属性可 用来处理 这些空白。

这段 文字拥有很多 空 白，空白可以由 一个 或多个 空格符、 制 表 符 或 换行符 产生，white-space属性可
用来处理 这些空白。

这段  文字拥有很多    空 白，空白可以由    一个
  或多个    空格符、   制      表      符      或 换行符
          产生，white-space属性可    用来处理 这些空白。

这段   文字拥有很多      空 白，空白可以
由    一个
  或多个      空格符、    制      表      符
      或 换行符
          产生，white-space属性可      用来处
理 这些空白。

这段 文字拥有很多 空 白，空白可以由 一个 或多
个 空格符、 制 表 符 或 换行符 产生，white-
space属性可 用来处理 这些空白。
```

图 9-40　white-space 各属性的效果(Opera 浏览器)

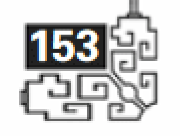
normal 情况下，所有空白都会被压缩成为一个空格，文字会根据元素大小自动产生换行；nowrap 使得文字不自动产生换行，除非遇到 br 元素；pre 将保留所有空白并按原样输出到浏览器，与 HTML 的 pre 元素作用相同，另外文字不会自动产生换行；pre-wrap 和 pre 的区别在于前者会产生换行；而 pre-line 和 pre-wrap 的区别在于前者会压缩空白。

2. direction

direction 属性可控制文本横向排版方式。ltr(left to right)表示从左向右排版，rtl(right to left)表示从右向左排版。注意这里指的排版是对于块级元素说的，也就是将文本作为整体进行从左向右或从右向左排版，文字内容的方向不受影响。如果想实现对元素中文字的控制，就要用到下面的属性。

3. unicode-bidi

该属性是 direction 的扩展，属性值 embed 表示本元素作为一个整体按照 direction 的属性



值进行排版，normal 表示不产生新的整体，bidi-override 表示按照 direction 的方式排版文本内容。该属性在 CSS 规范中叙述得比较晦涩，还是看一个具体示例吧：

```
p{
    font-family:"Times New Roman",serif;
    font-size:1.2em;
    direction:rtl;
    background-color:lightgrey;
}
span{
    background-color:black;
    color:white;
}

<p>The unicode-bidi property allows you to change the <span>direction of
embedded</span> text within text of a different direction.</p>
<p>The unicode-bidi property allows you to change the <span
style="unicode-bidi:embed;">direction of embedded</span> text within text of
a different direction.</p>
<p>The unicode-bidi property allows you to change the <span
style="unicode-bidi:bidi-override;">direction of embedded</span> text within
text of a different direction.</p>
```

图 9-41 为这段代码的效果。

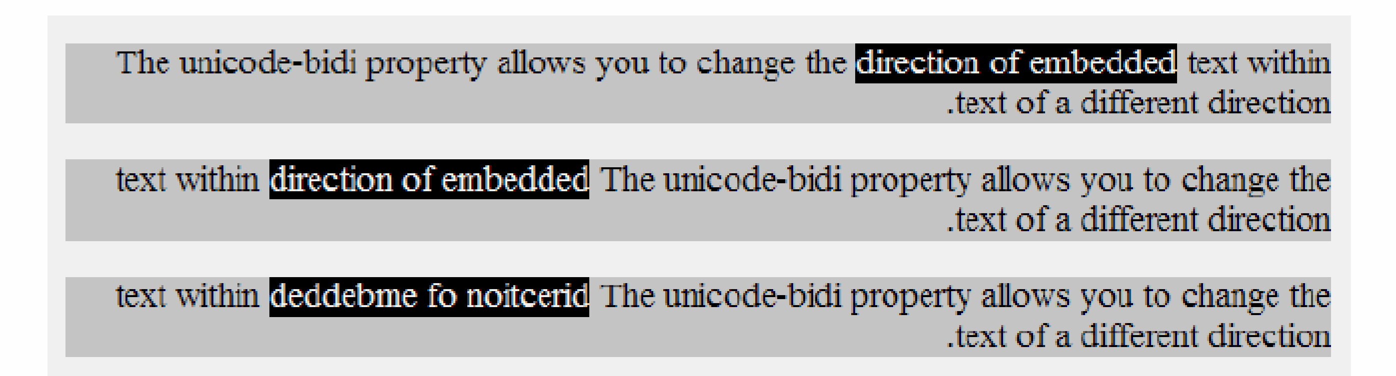


图 9-41　direction 和 unicode-bidi 属性

所有的 p 元素的 direction 属性为 rtl，从右向左排版。第一个 p 元素整体从右向左排版，但内部不受影响，文字均向右侧对齐，标点也移到了左边。第二个 p 元素中的 span 元素设置 unicode-bidi 属性为 embed，这表明 span 元素也要作为一个整体按照继承的 direction 属性值进行排版，因此 span 之前的部分移到了右侧，span 之后的内容移到了左侧。最后一个 p 元素的 span 的 unicode-bidi 属性为 bidi-override，表明 span 内的文字要按照其 direction 属性进行排版，因此 span 内的字符也都按照从右向左的方式进行书写。

4. text-overflow

尽管这个属性不是 CSS 规范中定义的，尽管只有 IE 浏览器支持这个属性，还是要向各位读者介绍它。有时候在固定区域内显示不下的文本会用省略号来替代，一般是通过程序判断来实现的。然而使用 text-overflow 属性可直接产生这种效果，减少了程序员的工作量。

请看下面的例子：



```
ul{
    font-size:12px;
}
li{
    width:130px;
    white-space:nowrap;
    border:1px solid lightgrey;
}
li#itemB{
    overflow:hidden;
    text-overflow:ellipsis;
}
li#itemC{
    overflow:hidden;
    text-overflow:clip;
}

<ul>
    <li id="itemA">即使超出了设定的边界也依然可以显示。</li>
    <li id="itemB">不显示超出的部分且在末尾添加省略号。</li>
    <li id="itemC">超出的部分被裁剪掉了不显示。</li>
</ul>
```

text-overflow 要配合 overflow 一起使用，如果 overflow 属性值不为 visible，text-overflow 的效果就会显现出来。white-space 设置为 nowrap 强制 li 中的文字不进行换行。text-overflow 有两个属性值 ellipsis 和 clip，分别运用在第二个和第三个列表中，图 9-42 为最终效果。

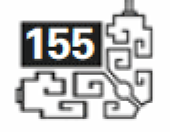
- 即使超出了设定的边界也依然可以显示。
- 不显示超出的部分且在...
- 超出的部分被裁剪掉了不

图 9-42　text-overflow 属性

5. layout-flow

该属性用来控制文本的排版方式，属性值 horizontal 表示横向从左至右排版，属性值 vertical-ideographic 表示从上到下排版，和中国古文的排版方式一致。该属性只有 IE 系列浏览器支持。请看：

```
h2, p{
    width:180px;
}
h2{
    text-align:right;
}
p{
    layout-flow:vertical-ideographic;
    line-height:1.5em;
}
```



```
<h2>青玉案&bull;元夕</h2>
<p>
东风夜放花千树。
更吹落、星如雨。
宝马雕车香满路。
凤箫声动，玉壶光转，一夜鱼龙舞。<br />
蛾儿雪柳黄金缕。
笑语盈盈暗香去。
众里寻他千百度。
蓦然回首，那人却在，灯火阑珊处。
</p>
```

图 9-43 中的文字从右向左，从上到下进行排版。

青玉案•元夕

东风夜放花千树。更吹落、星如雨。宝马雕车香满路。凤箫声动，玉壶光转，一夜鱼龙舞。蛾儿雪柳黄金缕。笑语盈盈暗香去。众里寻他千百度。蓦然回首，那人却在，灯火阑珊处。

图 9-43　使用 layout-flow 属性进行排版

延伸： CSS3 的文字模块 CSS Text Level 3 中提出了更多的有关文字的样式属性。感兴趣的读者可参考如下网址：
http://www.w3.org/TR/css3-text/

9.4　文字样式实战

A List Apart 是一家面向 Web 设计者的在线杂志网站，图 9-44 为该网站的一则新闻页面。现在就来分析一下该页面中部分内容使用了哪些与文字相关的 CSS 样式。

首先页面在 body 元素中设置了如下规则：

```
body {
    font: small Verdana, sans-serif;
    color: #333
}
```

字体大小为 small、字体族为 Verdana 和一个通用字体族 sans-serif，我们知道 Verdana 和 sans-serif 外观相近，笔划粗细均匀且不带有衬线装饰。颜色选用了一种深灰色且属于 Web 的安全色。



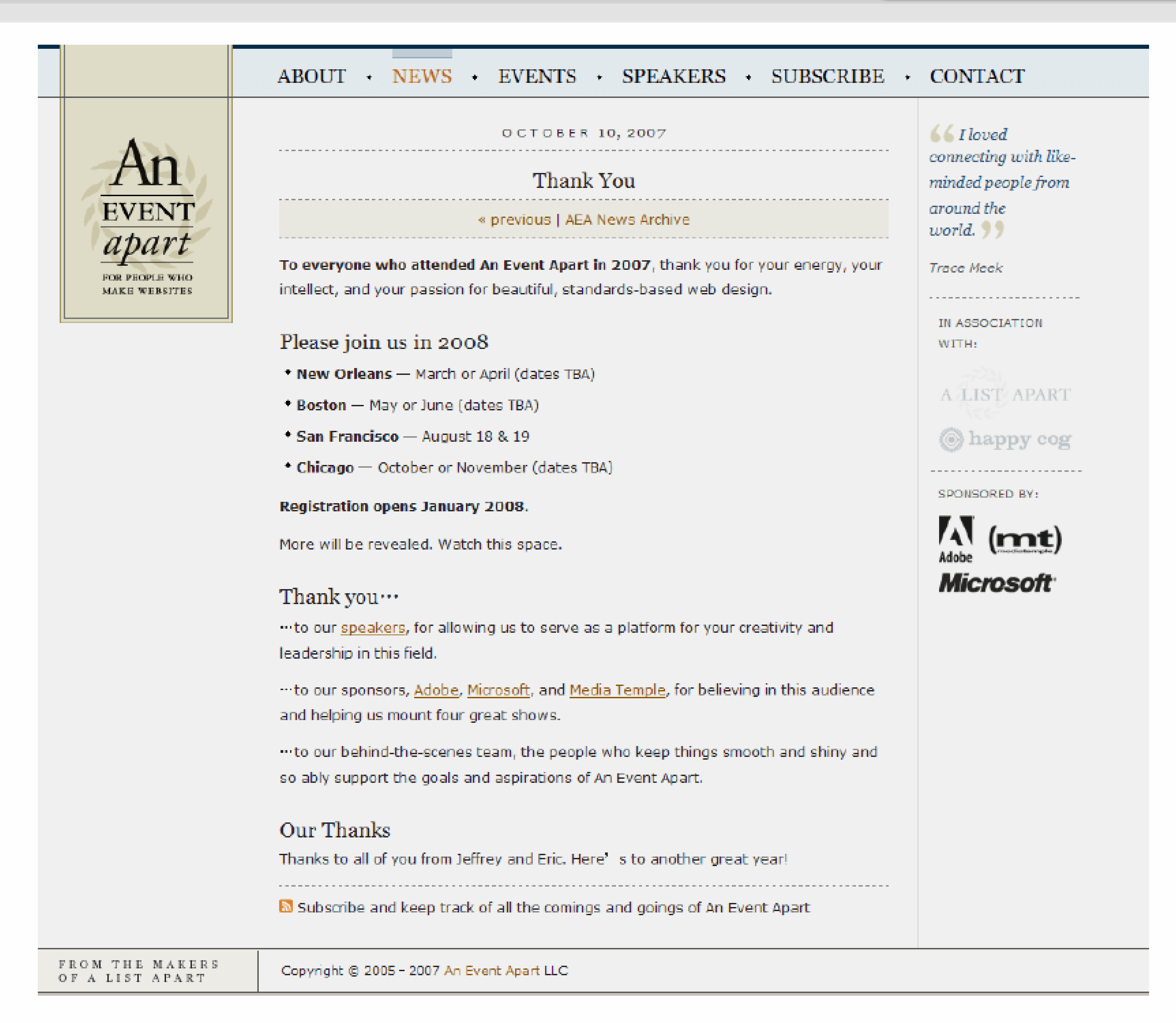


图 9-44　A List Apart 网站中的一则新闻页面

除了顶部的导航以外，页面所有内容被放置在 id 为 main 的 div 中，该 div 包含这样一些规则：

```
#main {
    font-size: 0.88em;
    line-height: 1.9;
}
```

那么页面中的文字大小的基准就是 small 大小的 0.88 倍，行距值是 1.9 且没有单位，因此各个子元素继承下来的行间距不是固定的计算值，而要用此值和自己的文字大小进行重新计算。这样就保证了子元素行距变化是动态的。

最上面的日期被放置在#dateline 元素中，并设置如下样式：

```
#dateline {
    padding-bottom: 0.9em;
    font: 0.85em Verdana, sans-serif;
    text-transform: uppercase;
    border-bottom: #999 1px dashed;
    letter-spacing: 0.33em;
    text-align: center
}
```

规则除了给 div 设定边框样式、填充外，还指定了字体大小为 0.85em，这是在 small 的 0.88 倍基础之上再进行计算的，text-transform 设为 uppercase，要求字母全部大写，最后设定字符间

距并让文本居中显示。

新闻标题被放在 h2 元素中，该元素中与文字相关的样式声明为：

```
h2{
    font: 1.6em Georgia, serif;
}
#content h2{
    text-align: center;
}
```

再来看正文的第一个段落，段落被放置在 p 元素中，其中粗体显示的部分被放在 strong 元素中，这样不需另设样式即可产生粗体效果。字体大小和行距都是继承父元素而得，分别是 0.88em 和 1.9。字体大小就是 small 的 0.88 倍，由于行距不带有单位，具体行距值需要用 1.9 乘以 0.88em 得出。

图 9-45 为上述部分在页面中的结构示意图，通过它可以更好地了解样式信息是如何组织和运用的。

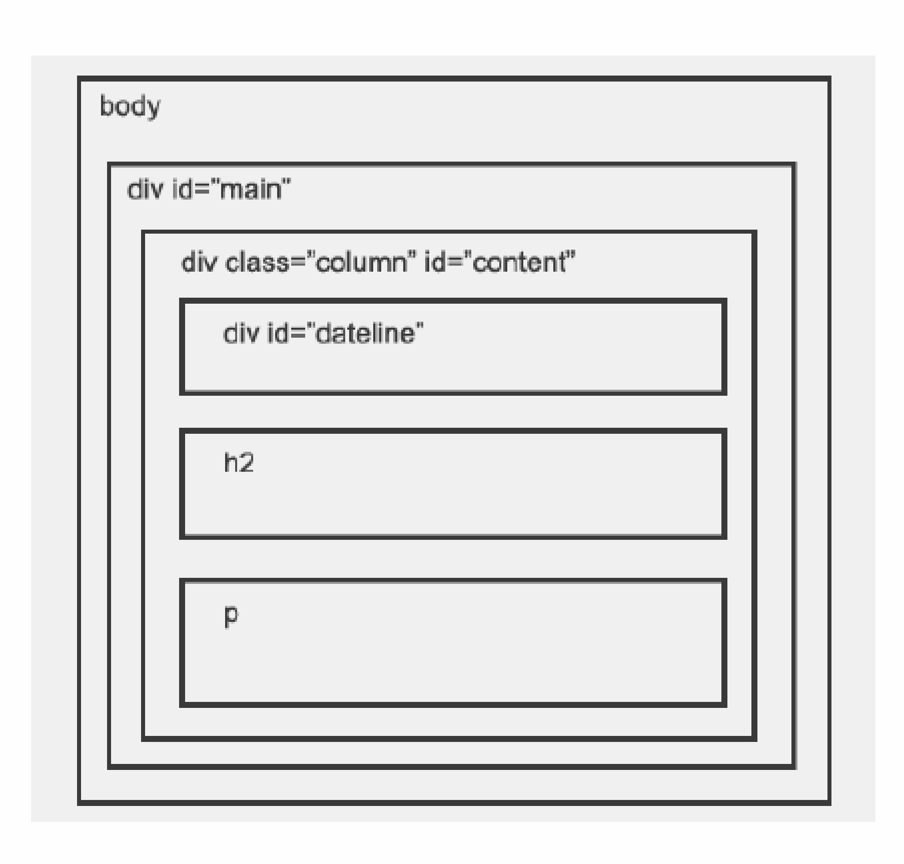


图 9-45　页面结构示意图

9.5　小　　结

本章讲解了如何使用 CSS 对字体、文本和段落进行样式控制。

字体族可以指定字体族名，最好在最后添加通用字体族以增强兼容性。在 Web 设计中字体尺寸通常使用相对大小进行设置，其中单位 px 和 em 比较常见。

CSS 还提供了多种用于控制文本和段落的样式属性，它们可用来装饰文字、设定行间距、设置水平和垂直方向的对齐方式、段落的排版方式等。

在本章的实例部分中，分析了一个 A List Apart 网站的页面，看该页面是如何组织和运用与字体样式、段落排版相关的样式的。

下一章将学习有关 CSS 对页面链接进行样式控制的内容。

第 10 章　链 接 处 理

链接是 Web 页面的重要组成部分，没有链接的页面也就几乎失去了 Web 页面存在的意义。用户通过链接可访问网络中的其他资源。CSS 通过伪类来设置不同状态下链接的样式，本章将介绍实现超链接的锚元素及其不同状态，进而介绍如何运用 CSS 伪类为链接添加样式。

本章主要内容

- 链接和锚元素
- 链接的状态
- 如何给链接添加样式
- LVHA 规则

10.1　认 识 链 接

10.1.1　锚元素

Web 页面正是通过链接实现在不同页面之间跳转、访问不同网络资源的目的。(X)HTML 提供的锚(Anchor)元素可用来设置超链接(Hyperlink)，实现对资源的访问。一般一个锚元素包含 href 属性，属性值是一个 URL，通过单击这个锚元素就可以访问所指定的 URL。

锚元素的标记形式为<a>，下面是一个链接的示例：

```
<a href="http://www.huistd.com/">欢迎访问工作室网站</a>
```

注意：a 元素中不可以再包含其他的 a 元素。

10.1.2　链接状态

首先，浏览器可以区分未访问过的和已经访问过的(浏览器的历史记录会保存已访问过的页面信息)链接，a 元素必须包含 href 属性以指明访问的地址。CSS 提供两个链接伪类——:link 和:visited，用来区分这两种状态。下面的代码就可以控制未访问和已访问的链接的文字颜色：

```
a:link{color:red;}          /* 未访问过的链接文字颜色为红色 */
a:visited{color:blue;}      /* 访问过的链接文字颜色为蓝色 */
```

CSS 还提供了三个动态伪类——:hover、:active 和:focus，hover 表示当鼠标悬停在链接上时的状态，active 表示当在链接上单击鼠标时的状态，即激活状态，focus 表示链接获得焦点时的状态，尽管 IE 中的链接可以获得焦点，但是它不支持这个 CSS 伪类，因此本章不涉及 focus 这个状态。

注意： 为了简单起见，本书使用伪类的名称来描述对应的锚元素所处的状态。例如“hover 状态”表明有鼠标悬停在该锚元素上。

10.1.3 默认效果

浏览器会给链接的不同状态提供默认的样式，用 IE 浏览器打开包含如下代码的页面：

```
<a href="http://www.huistd.com/">未访问过的链接</a><br />
<a href="about:blank">已访问过的链接</a>
```

由于 IE 浏览器启动后默认打开空白页，浏览器的地址栏会显示 about:blank，因此使用 about:blank 作为 URL 可以表示已访问过的链接。如果 IE 不是以空白页作为默认页打开的，读者可亲自单击第二个链接，然后再退回来，这时该链接就变成已访问过的了。我们将看到如图 10-1 所示的效果，链接文字有下划线装饰，未访问的链接文字颜色为蓝色，访问过的链接文字颜色为暗紫红色。

未访问过的链接
已访问过的链接

图 10-1　链接默认样式：访问前(上)和访问后(下)状态

当把鼠标放在链接上或在链接上单击时，链接样式不发生改变，唯一变化的是鼠标的图标变成了手型，如图 10-2 所示。

悬停状态的链接　激活状态的链接

图 10-2　链接默认样式：悬停(左)和激活(右)状态

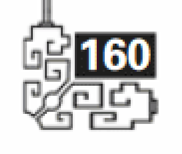
10.2 添加链接样式

10.2.1 LVHA，爱和恨

不要被这一节的标题吓到了，暂时不懂这是什么意思也不要紧，学习完本节内容后你就会明白它的含义。

前面说过链接有 4 个状态：link、visited、hover 和 active，注意到它们出现的顺序了吗？看看表示 4 个状态的英文单词的首字母分别是什么？对了，就是 LVHA，当我们给链接不同的状态添加样式时，一定要按照这个顺序，否则某些状态的样式将不会生效。你也可以把这个样式顺序记为爱和恨(LoVe and HAte)。

下面的代码分别为链接的 4 个状态设置不同的颜色，注意它们的顺序：



```
a:link {color:red;}
a:visited {color:blue;}
a:hover {color:yellow;}
a:active {color:purple;}
```

之所以这样设定它们的顺序，是因为 CSS 的层叠规则的影响。4 个选择符都只包含一个伪类，因此它们的确定度是一致的，只能通过出现顺序来确定优先级别。假如用户把鼠标移到链接上，浏览器就会按照顺序查找第一个满足当前状态的 CSS 样式，这时按钮处于 hover 状态，那么最后出现的:active 伪类不起作用，而它之前的:hover 伪类选择符将会匹配当前的链接，从而达到更改样式的目的。假如我们这样修改上面规则出现的顺序：

```
a:hover {color:yellow;}
a:link {color:red;}
a:active {color:purple;}
a:visited {color:blue;}
```

如果该链接已经被访问过，那么最后一条规则的优先级最高，它总会匹配当前链接，因此无论当鼠标经过链接还是单击链接，伪类:hover 和:active 中设定的样式均不会起作用。所以一定要记住这个顺序。

10.2.2　下划线

网页上最常见的链接样式就是下划线了，这也是浏览器默认的样式，目的是把链接文字和普通文字区分开来。有时候我们习惯于将普通状态链接的下划线取消，采用其他方式区别于普通文字(比如加粗、更换颜色等)，当鼠标悬停在上面时再显示下划线。具体做法如下：

```
a{color:#00FF33;};                           /* 所有状态的链接为绿色 */
a:link, a:visited{text-decoration:none;}     /* 去掉下划线 */
a:hover, a:active{text-decoration:underline;} /* 添加下划线 */
```

遗憾的是通过 text-decoration 属性添加的下划线样式稍显简单，只能是 1px 大小的实心线条，那么能不能添加其他样式的下划线呢？我们现在就来介绍另外两种向链接文字添加下划线的方法。

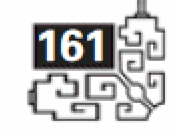
还记得盒模型的概念吗？每个(X)HTML 元素都可以抽象为一个盒子，通过指定盒子的下边框样式就可以达到添加下划线的效果。请看下面的示例。

(X)HTML 代码：

```
单击<a href="http://www.huistd.com/">这里</a>访问工作室网站。
```

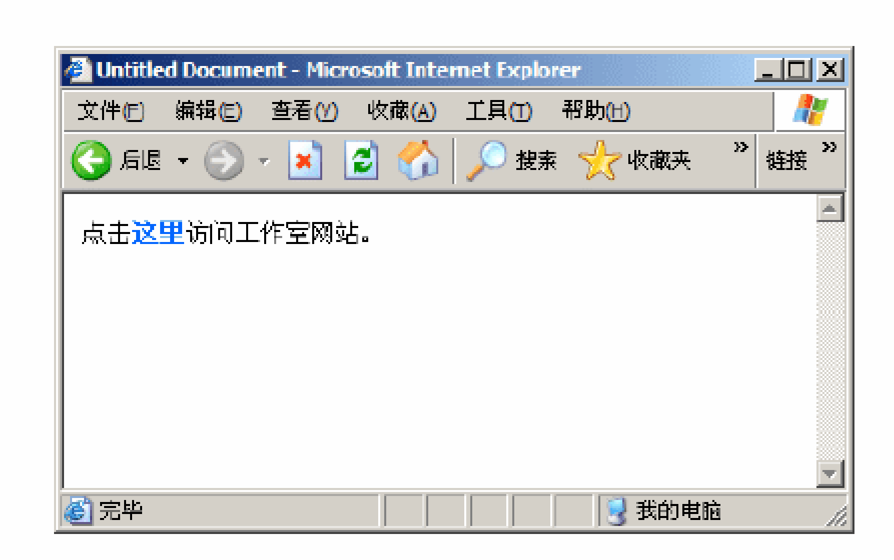
CSS 代码：

```
body{
    font-size:14px;
}
a{
    font-weight:bold;
    text-decoration:none;
```



```
}
a:link, a:visited{
    color:#0066FF;
    border-bottom:1px dotted #0066FF;
}
a:hover{
    color:#00FFFF;
    border-bottom:1px dotted #00FFFF;
}
```

我们首先设定字体大小为 14px，然后设置链接文字为粗体并且把下划线全部去掉，最后分别设置 link、visited 和 hover 状态的颜色和下边框的颜色。当分别用 IE 和 Firefox 浏览器查看这个页面时，会发现 IE 中并没有出现预期的虚线状的下划线(见图 10-3)。



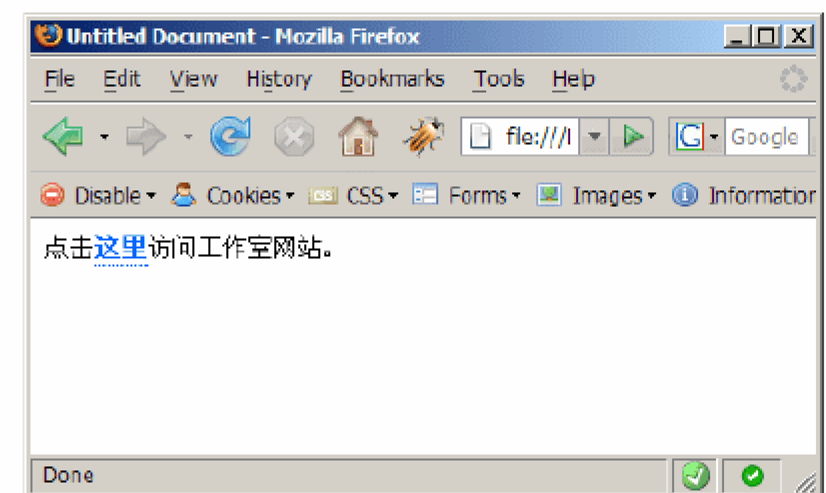


图 10-3 IE(左)没有下划线，Firefox(右)有下划线

为了让 IE 能够显示下划线，我们需要改变 a 元素的显示方式为内联块类型(inline-block)，这样“盒子”的下边框才能显示出来，a 元素的样式重写为：

```
a{
    font-weight:bold;
    text-decoration:none;
    display:inline-block;
}
```

从图 10-4 中可见，IE 显示了下划线。

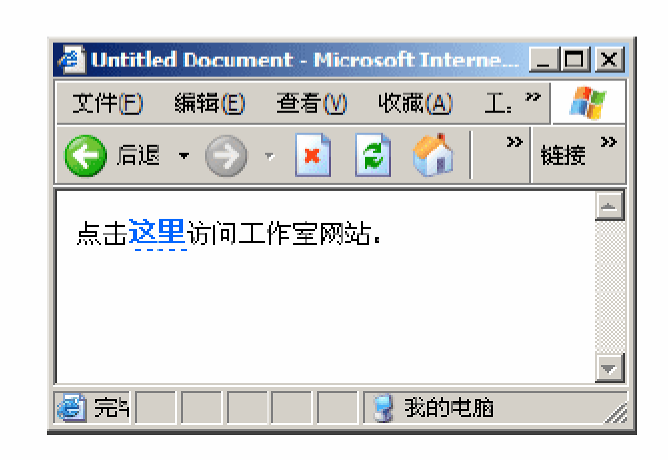


图 10-4 修改后，IE 可以显示下划线

个性化十足的设计师可能还对这个样式不满意，完全掌控下划线样式才能满足他们的需求，现在我们就来介绍如何使用自定义的图片完成下划线的制作。首先使用图形图像工具(比如 Photoshop)创建一个 4 像素宽 2 像素高的 gif 图片(见图 10-5)，命名为 underline.gif。



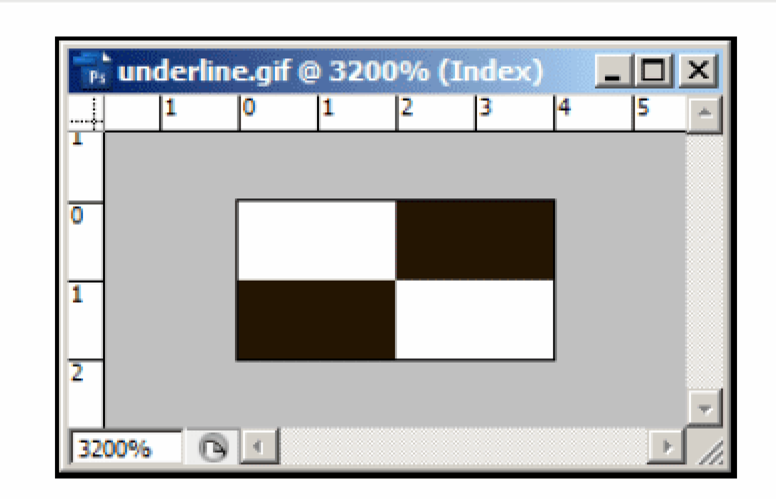


图 10-5　自制的下划线图像

(X)HTML 代码不变，这里更改 CSS 代码：

```
body{
    font-size:14px;
}
a{
    color:#0066FF;
    font-weight:bold;
    text-decoration:none;
    background: url(images/underline.gif) repeat-x left bottom;
}
```

这里我们不考虑链接的各个状态，让所有状态的样式一致，重点在于如何使用图片做下划线装饰。background 属性为 a 元素添加了一个背景图片 underline.gif，并让它在 x 方向上重复，最后设置对齐方式为左对齐和底端对齐。有关 background 属性的详细用法会在下一章进行介绍。图 10-6 为所取得的效果。

点击**这里**访问工作室网站。

图 10-6　使用背景图片添加下划线

10.3　链 接 实 战

链接除了以文字的形式存在外，还可以经过修饰，成为一个按钮，如果为链接添加一些背景图片，还可以达到更好的效果。尽管使用其他元素也可以制作按钮，但是按钮的功能实现还要依靠添加 JavaScript 一类的脚本代码，而锚元素本身就包含了单击事件，因而实现起来更加简单。这里将向读者介绍以两种不同的方式来创建按钮。

10.3.1　简单方式

现在我们使用一种简单的方式来创建按钮，按钮的边框用元素的边框绘制，颜色用元素背景绘制，请看如下示例。

(X)HTML 代码只有一行：

```
<a href="http://www.huistd.com/">按 钮</a>
```



CSS 代码：

```
a{
    text-decoration:none;
    width:80px;
    height:22px;
    text-align:center;
    padding-top:7px;
    font-size:12px;
    font-weight:bold;
    display:block;
}
a:link, a:visited{
    border:solid 1px #66777A;
    background-color:#90DAE6;
    color:#09505B;
}
a:hover{
    background-color:#E9FCFF;
    color:#09505B;
}
a:active{
    background-color:#13434B;
    color:#E9FCFF;
}
```

我们去掉 a 元素所有状态下的下划线，设置按钮的宽度和高度，让文字显示在按钮中间，设定适当的上边距让文字达到垂直居中的效果，设置字体大小和粗细程度，最重要的是要把 display 属性设置为 block，这样高度和宽度的设置才会生效。

然后定义不同状态的样式，通过背景色和文字颜色来进行区分。如图 10-7 所示为按钮的三个状态的不同样式。



图 10-7　按钮的三个状态

10.3.2　图像方式

图 10-8 为网易邮箱界面的一部分，其中收信和写信按钮就是使用锚元素加图片制作的。当鼠标悬停在上面时，按钮会更换图片以另外一种样式展现出来，但是这样的方法也有个小小的缺陷，那就是当单击鼠标时，锚元素获得焦点后会有个点状的线框出现。





图 10-8　163 邮箱页面

下面我们就来制作类似这种效果的按钮，首先用图像软件制作好表示 3 个状态的图片(我们让 visited 和 link 的状态一致)，把 3 幅图保存为一张 PNG 格式的图片 bgButton.png，如图 10-9 所示。



图 10-9　表示 3 种状态的按钮图片，由上到下依次是 link、hover 和 active

(X)HTML 代码相当简单，仍然只有一行：

```
<a class="imageButton" id="buttonA" href="http://www.huistd.com"><span>按
 钮</span></a>
```

CSS 代码如下：

```
a.imageButton{
    text-decoration:none;
    width:100px;
    height:40px;
    display:block;
}
a.imageButton span{
    font-weight:bold;
    visibility:hidden;
}
a#buttonA:link, a#buttonA:visited{
    background:url(images/bgButton.png) no-repeat 0 0;
}
a#buttonA:hover{
    background:url(images/bgButton.png) no-repeat 0 -40px;
}
a#buttonA:active{
    background:url(images/bgButton.png) no-repeat 0 -80px;
}
```

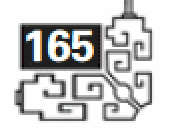


图 10-10 为最终效果，当鼠标悬停在按钮上或单击按钮时，按钮会以不同的样式呈现。注意到我们给 a 元素指定了一个 class 名和一个 id 名，class 名表明该 a 元素是用来做图片按钮的，因此该类的样式去掉了下划线，并设置了高度和宽度，不要忘记把 a 元素改成块(block)级别的。接下来的 CSS 代码可能会让读者产生疑问，为什么写个 span 元素却还要设置其 visibility 属性为 hidden，把它隐藏掉呢？写这个 span 元素的目的在于使(X)HTML 代码本身包含按钮的信息，有了这个信息，搜索引擎才可能搜索到这个关键字。如果用户的浏览器不支持 CSS，那么这个链接将会以纯文本的方式显示，不会对访问产生影响。最后，通过 id 和伪类选择符，我们分别让背景图在 a 元素的三个状态下进行位置变换(有关背景图像属性的用法请参考第 11 章)，从而产生了如图 10-10 所示的效果。



图 10-10 使用 a 元素制作按钮

图 10-11 展示了变换背景图位置的原理。

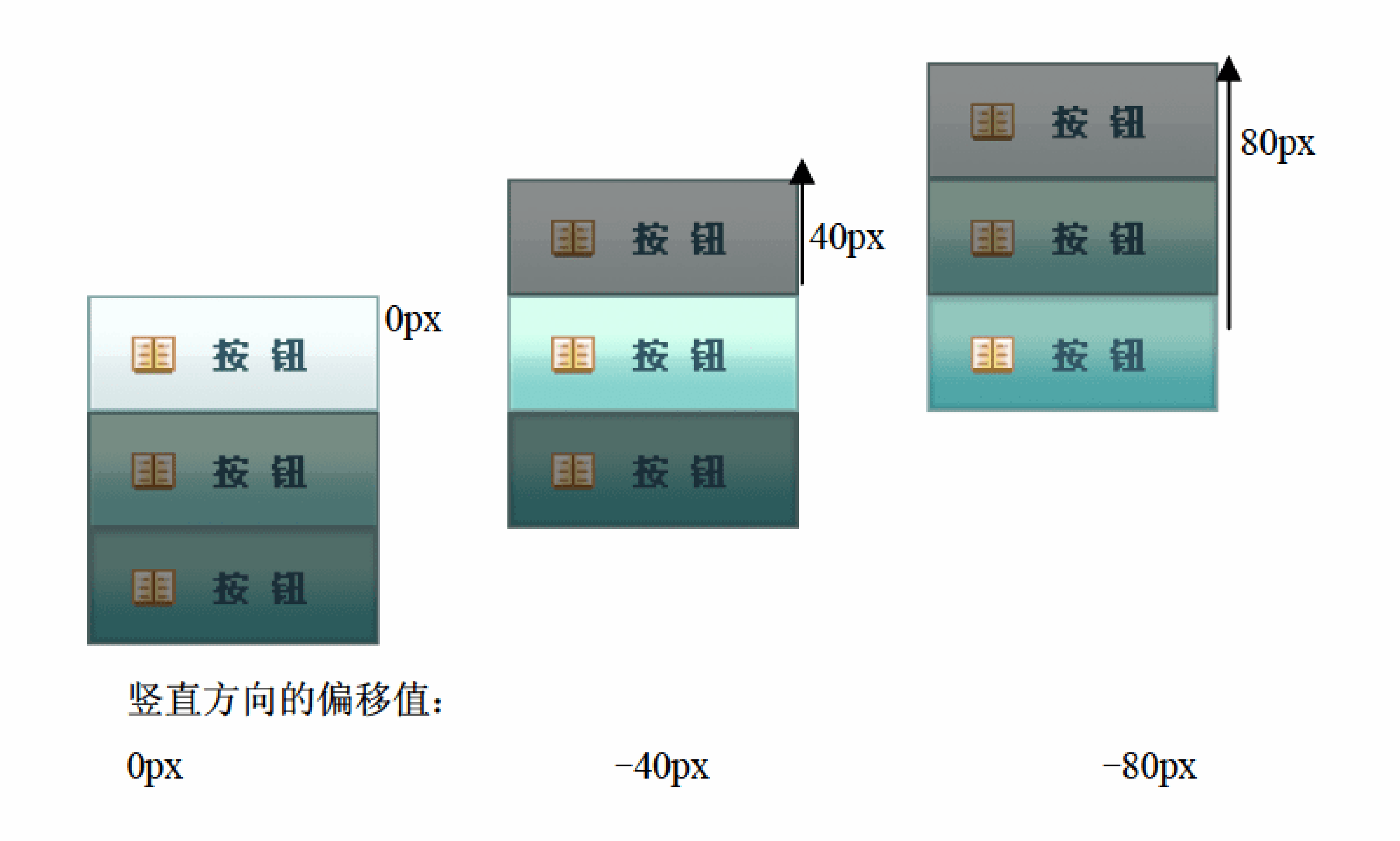
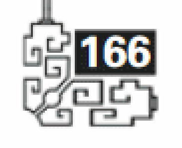


图 10-11 背景图位置变换的原理

这样，一个完全由 CSS 制作的按钮就完成了，不用添加任何脚本代码就能实现按钮的基本功能。本例中将 class 和 id 分开的好处可能不是很明显，假如页面有很多大小一致图片按钮，通过类选择符就可以一次设置好所有 a 元素的大小和显示方式，然后再通过 id 选择符分别为每个的按钮添加不同的图片。

功能性的按钮通常没有必要区分访问和未访问这两个状态，因为用户要经常使用按钮提供的功能，而不关心这个链接是否已经访问过，因此本小节示例中全部让这两个状态的样式一致。



提示：我们将表示按钮三种不同状态的图像放置在一张图像上，通过位置变换获取相应的图像。这样做能有效地减少浏览器对 Web 服务器的资源请求，从而加快页面的下载速度。

前面介绍过的两种制作按钮的方法有个缺陷，按钮的尺寸由 CSS 规定死了，不能按照按钮上的文字个数灵活变化宽度，如果按钮字数过多还会出现显示不下的情况。为了解决这个问题，我们不能使用尺寸固定的图片作为背景，而是使用可以重复出现的(见图 10-12)，命名为 bgButtonEx.png。

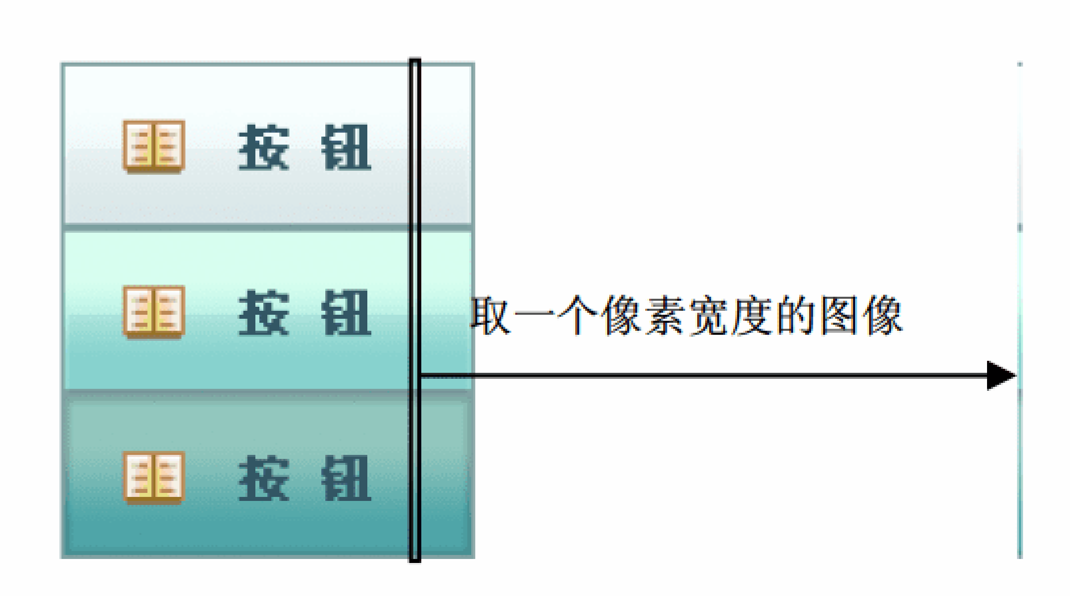


图 10-12　重新制作背景图片

按钮左边的小图标也要单独提取出来，取名为 dec.gif。为了能融合在不同的背景中，图像需要做透明处理(见图 10-13，灰白格表示图像的透明部分)。

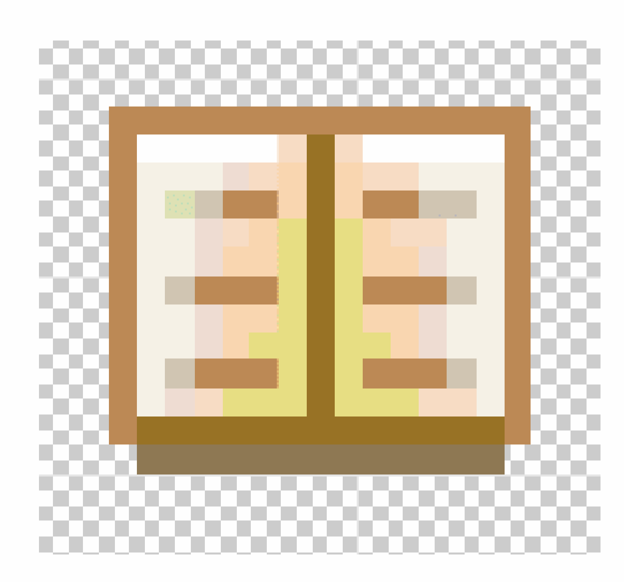


图 10-13　提取按钮上的图片，并让背景透明

前期工作已经完成，该到写代码的时候了，(X)HTML 代码保持不动，为了展示按钮的自动伸缩功能，我们再添加一个新按钮，并指定 id 值为 buttonB：

```
<a class="imageButton" id="buttonA" href="http://www.huistd.com"><span>按
 钮</span></a>
<a class="imageButton" id="buttonB" href="http://www.huistd.com"><span>四
 字 按 钮</span></a>
```

现在重写 CSS 样式代码，为了使读者能进一步掌握 CSS 编写技巧，这里将分步进行。

首先，将设定宽度的代码取消，并设置文字大小为 12px，将颜色设置成原图像中的文字颜色。接着让 span 元素显示出来，取消原来的 visibility:hidden 声明。最后更改按钮各个状态的背景图为 bgButtonEx.png，并让它在水平方向上重复出现。

代码如下：

```
a.imageButton{
    text-decoration:none;
    height:40px;
    display:block;
    font-size:12px;
    color:#315664;
}
a.imageButton span{
    font-weight:bold;
}
a#buttonA:link, a#buttonA:visited, a#buttonB:link, a#buttonB:visited{
    background:url(images/bgButtonEx.png) repeat-x 0 0;
}
a#buttonA:hover, a#buttonB:hover{
    background:url(images/bgButtonEx.png) repeat-x 0 -40px;
}
a#buttonA:active, a#buttonB:active{
    background:url(images/bgButtonEx.png) repeat-x 0 -80px;
}
```

图 10-14 为修改后的效果。

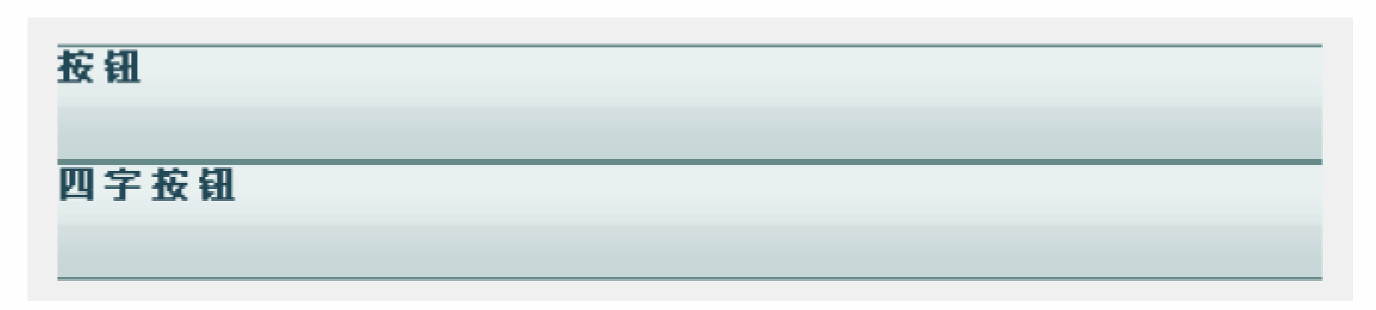


图 10-14　修改代码之后的按钮

从图 10-14 看出，按钮的宽度是随着浏览器宽度变化的而不是根据文字宽度变化，这是块级元素的特性，它会占满一整行。另外按钮中的文字位置也不合适。为了解决这两个问题，我们再向 a.imageButton 中添加如下两条规则，并修改 height 属性值：

```
height:25px;
float:left;
padding:15px 20px 0 20px;
```

图 10-15 为添加规则后的效果。



图 10-15　添加规则后效果

由于让元素向左浮动，按钮不会再占满一整行，按钮宽度可以根据文字多少而变化了。添加的 padding 属性使得按钮中的文字与边框之间形成一定距离，由于上边距会影响整个高度，因此将 height 值改为 25px(40–15=25)。现在按钮两边还缺少边框，小图标也没有添加，于是我



们再次添加新规则，整体代码应为：

```
a.imageButton{
    text-decoration:none;
    height:25px;
    padding:15px 20px 0 12px;
    display:block;
    float:left;
    font-size:12px;
    color:#315664;
    border-left:1px solid #7A9C9C;
    border-right:1px solid #7A9C9C;
}
a.imageButton span{
    font-weight:bold;
    background:url(images/dec.gif) no-repeat 2px 0px;
    padding-top:1px;
    padding-left:30px;
}
a#buttonA:link, a#buttonA:visited, a#buttonB:link, a#buttonB:visited{
    background:url(images/bgButtonEx.png) repeat-x 0 0;
}
a#buttonA:hover, a#buttonB:hover{
    background:url(images/bgButtonEx.png) repeat-x 0 -40px;
}
a#buttonA:active, a#buttonB:active{
    background:url(images/bgButtonEx.png) repeat-x 0 -80px;
}
```

图 10-16 为最终效果，一个宽度自适应的按钮就做好了。读者是不是已经领略到 CSS 的魅力了？我们没有修改任何(X)HTML 代码，而只调整了样式就能达到新的效果，是不是很神奇？

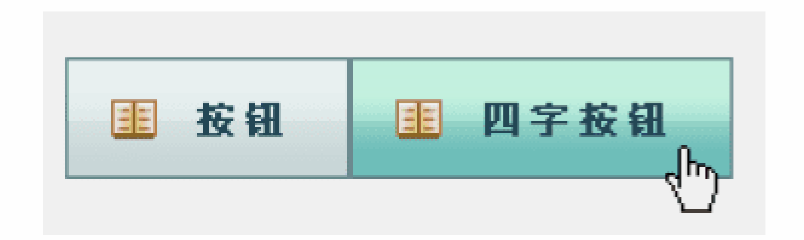


图 10-16　宽度自适应按钮的最终效果

10.4　小　　结

本章讲述了如何给 Web 页面中链接添加样式以及利用锚元素固有的特性来实现按钮。

(X)HTML 文档使用锚元素表示链接，链接分为 5 个状态：正常、已访问、悬停、激活和获得焦点。CSS 提供了 5 种伪类，分别对应不同状态的锚元素，它们是:link、:visited、:hover、:active 和:focus。由于 IE 浏览器不支持:focus 这个伪类，因此实际只使用前 4 个伪类。

对应不同状态的 CSS 规则要满足一定的顺序，顺序是由 CSS 层叠规则决定的，否则一些

状态的样式会与代码不符。这个顺序就是:link、:visited、:hover、:active，将它们的首字母提出来缩写成 LVHA 会更便于记忆。

由于锚元素本身包含了多种状态，也能自动地响应鼠标单击，因此用它来实现按钮是个不错的主意。只要添加少量的样式代码，再设置背景色或添加一些背景图片，按钮就可以轻松地实现了。

链接还常与列表元素结合在一起使用，这是构建网站导航的主要方式之一，有关导航菜单的设计和制作问题将会在第 12 章中讲解。

下一章将介绍图像和背景的相关知识。

第 11 章　图像和背景

很难想象缺少图像的网站会是什么样子。与文字内容相比，图像提供的信息更直观，而且一些网站就是依靠优秀的图形图像设计创造出非凡的视觉效果的，令访问者过目难忘。网页中常见的图像使用方式有页面背景图、网站标志、网页插图和一些起装饰作用的小图标等。

CSS 不仅提供了控制图像的样式属性，还支持在样式代码中插入和调整图像。本章将会介绍如何利用 CSS 的相关样式属性对图像进行处理，此外，还会介绍一些图像的基本知识以及如何制作适合 Web 环境的图像。

本章主要内容

- Web 中常见的图像格式
- img 元素的使用
- 图文混排技术
- 背景属性和背景图像的使用

11.1　图 像 格 式

GIF、JPEG 和 PNG 是 Web 页面最常使用的三种图像格式，它们拥有不同的特性和适用范围。本节将介绍有关三种图像格式的基本知识，帮助设计者根据需要选择合适的图像格式。有关图像文件的具体格式、存储结构和压缩算法等内容将不做介绍。

11.1.1　GIF

GIF，英文全称为 Graphic Interchange Format，即可交换图像格式，1987 年由 CompuServe 公司提出并开发，是早期 Web 浏览器所显示的第一种图像格式。目前 Web 页面上仍然大量地使用着 GIF 图像。

GIF 采用索引色表来存储图像的颜色信息，最大可包含 8 位的颜色信息，即 256 种颜色(2^8=256)。位数越少则图像的颜色也越少，文件体积也较少。简单地讲，GIF 并不单独存储图像中每个像素的颜色值，而是把图像的所有颜色组织成一个颜色表，每个像素只包含对这个颜色表的引用，既使图像分辨率很大，由于颜色数固定，只多添加了一些像素的引用，文件尺寸也不会很大。如图 11-1 所示为两幅拥有不同颜色数量的 GIF 图片，可以看出当颜色数量较少时，画面比较粗糙，呈现出一种颗粒状。

GIF 支持透明显示，透明程度只有两个级别，即透明和不透明，没有任何半透明的效果。透明区域下面的任何东西(通常是页面背景色或其他图像)都可以显示出来。几乎所有的浏览器都支持 GIF 透明效果。另外，一张 GIF 图片可以包含多个帧，它们连续显示形成动画，多数浏览器也支持 GIF 动画。



256 色 94.0KB



32 色 46.3KB

图 11-1 不用颜色数量的 GIF 图像

通常，色彩数量较少、需要透明处理或以动画显示的图像使用 GIF 格式保存，比如页面背景图案，企业的标识，一些广告动画等。

11.1.2 JPEG

JPEG 是 Joint Photographic Experts Group 的缩写，译为联合图像专家组。JPEG 格式的图像文件通常以.jpeg 或.jpg 作为后缀，因而也有人称之为 JPG 图像。

JPEG 包含 24 位 RGB 颜色信息，比 GIF 支持更多的色彩数量，能表现更丰富的色彩变化和图像细节，与 GIF 不同之处在于 JPEG 不使用颜色表存储色彩信息。JPEG 使用了一种有损耗的压缩算法，图像信息会因此而产生丢失，如果压缩比率不是很高的话，肉眼是无法分辨损耗的。在使用图像处理软件保存 JPEG 格式时会选择压缩质量，通常是 0~100(Photoshop 提供的范围是 0~12)。数值越低则图像质量也越低，但是文件体积较小。图 11-2 展示了两幅质量不同的 JPEG 图像。



较高品质 172KB

较低品质 36.4KB

图 11-2 不同压缩质量的 JPEG 图像

由于 JPEG 可以保存 24 位的颜色信息，那些色彩丰富的图像可考虑使用该格式保存，比如人物肖像、产品展示、风景绘画等。JPEG 对于颜色数量较少的图像压缩效果并不好，会让颜色“变脏”。

11.1.3　PNG

PNG，全称 Portable Network Graphics，即可移植网络图像。PNG 是专门针对 Web 使用而设计的，它支持 8 位调色板(与 GIF 类似)，支持 16 位灰度和 48 位真彩色。采用无损耗压缩策略，可包含 8 位或者 16 位透明信息的 alpha 通道，这就是说比起 GIF 只有透明和不透明两种效果，PNG 可以支持多至 65000 个不同程度的透明，但 256 个透明程度的更为常见。图 11-3 对比了含有透明信息的 GIF 与 PNG 图像的区别。目前，PNG 已经成为 W3C 的推荐标准(http://www.w3.org/Graphics/PNG/)。




图 11-3　GIF(左)与 PNG(右)图像的透明效果对比，灰白格表示图像透明部分

尽管 PNG 有上述这些优秀的特性，但它并没有得到广泛的支持，当作为 img 元素插入到网页中时，IE6 根本无法正确显示透明效果，好在这一问题已在 IE7 中得到修正(见图 11-4)。但 Windows 平台的 IE 浏览器至今仍然无法显示 CSS 中设定的带有透明信息的 PNG 图像。

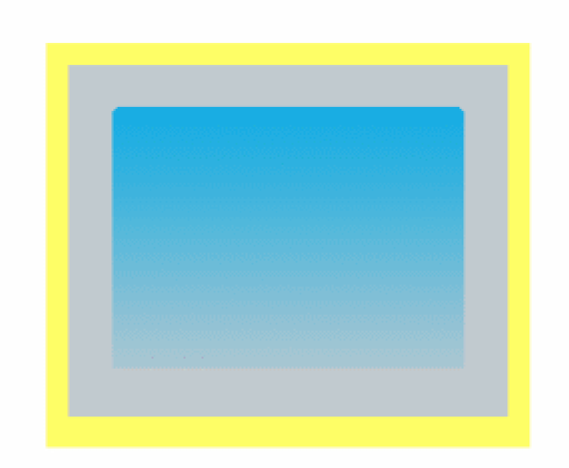
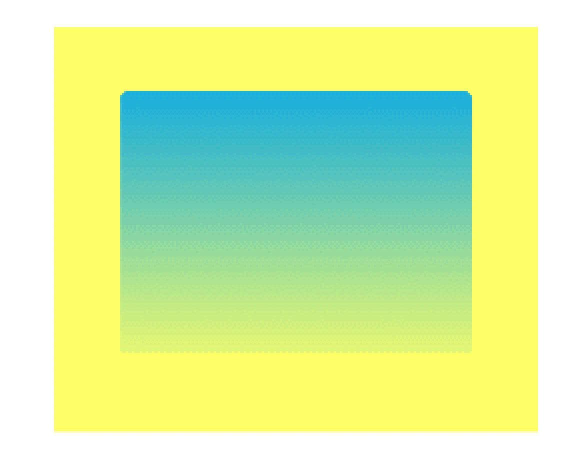
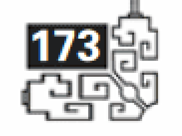



图 11-4　一张 PNG 图像放置在黄色背景上(IE6(左)将透明部分显示为灰色，IE7(右)可以正常显示透明效果)

11.2　给图像添加样式

本节所涉及的图像是指通过<img>标记插入的，作为(X)HTML 文档的一部分。CSS 并没有专门的样式来控制图像显示，你可以使用已经学过的部分样式属性来添加效果，其中与盒模型相关的样式属性经常会被使用。在介绍添加样式之前，我们先来熟悉一下 img 元素。



11.2.1　img 元素

img 元素可以将外部的图像资源插入到 Web 文档中，属于内联元素，不会引起任何换行，也不会产生多余的空白，图像底边与周围的文字的基线位置对齐。比如下面的代码会产生如图 11-5 所示的效果：

```
<p>安装 Firebug 后，浏览器的导航工具条中会多一个<img src="images/firebug_logo.gif"
alt="Firebug logo" />图标，点击它就可以启动 Firebug 了。</p>
<p>After installing Firebug there is an icon <img src="images/firebug_logo.gif"
alt="Firebug logo" /> appeared in the navigation toolbar. Clicking this icon
will launch the Firebug.</p>
```

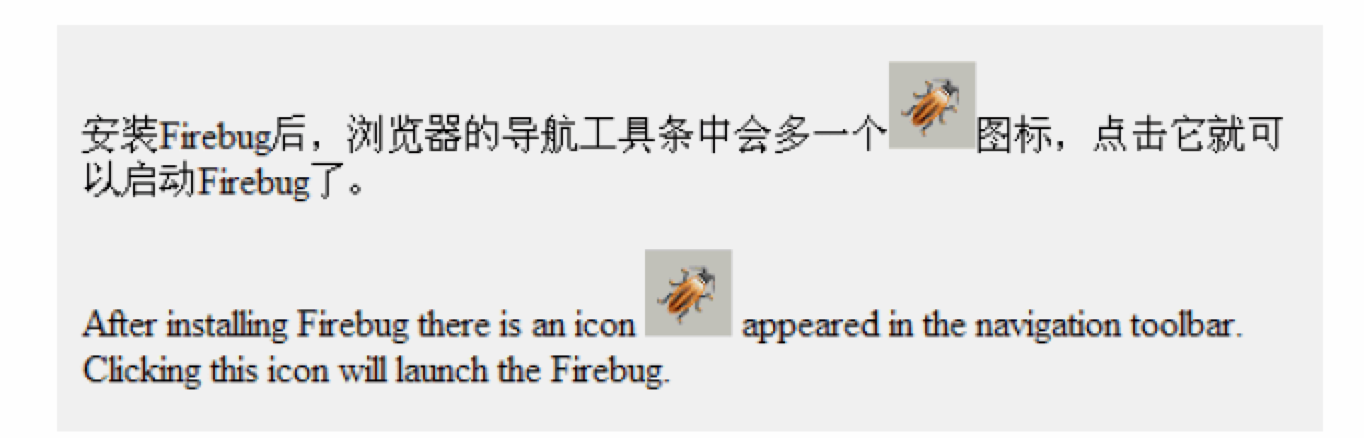


图 11-5　插入图像后的默认样式

注意： 插入的图像是与西文字符的基线对齐的，基线位置比中文字符的底边略高一些，这也是图 11-5 中图像底边会比周围的中文字符底边高一点儿的原因。另外，XHTML 要求 img 元素必须要有 alt 属性，以便在图像加载完成之前或者加载失败时能够以文字的形式向用户提示图像的内容或作用。

当图片被放置在链接中时，浏览器会在图像周围显示一个 2px 宽的蓝色边框，这个蓝色就是链接文字默认的颜色。大部分情况下设计者都会将蓝色边框去掉，你可以将 img 元素的 border 属性置为 0 或者修改 CSS 的 border 属性。

img 元素中包含了一些控制图像位置和样式的属性，比如 align、valign、hspace 和 vspace 等，本书将不使用这些属性，把控制样式的任务全部交给 CSS 来完成，(X)HTML 只负责内容的安排，达到各尽其职的目的。

11.2.2　盒模型相关样式

与盒模型相关的几个样式属性都可以用到图像上。比如，用 border 去掉链接图像默认的蓝色边框或者添加自定义的边框，用 padding 添加边框与图像之间的空白，用 margin 添加图像与其他元素之间的距离。图 11-6 展示了对图像添加不同样式声明的效果。



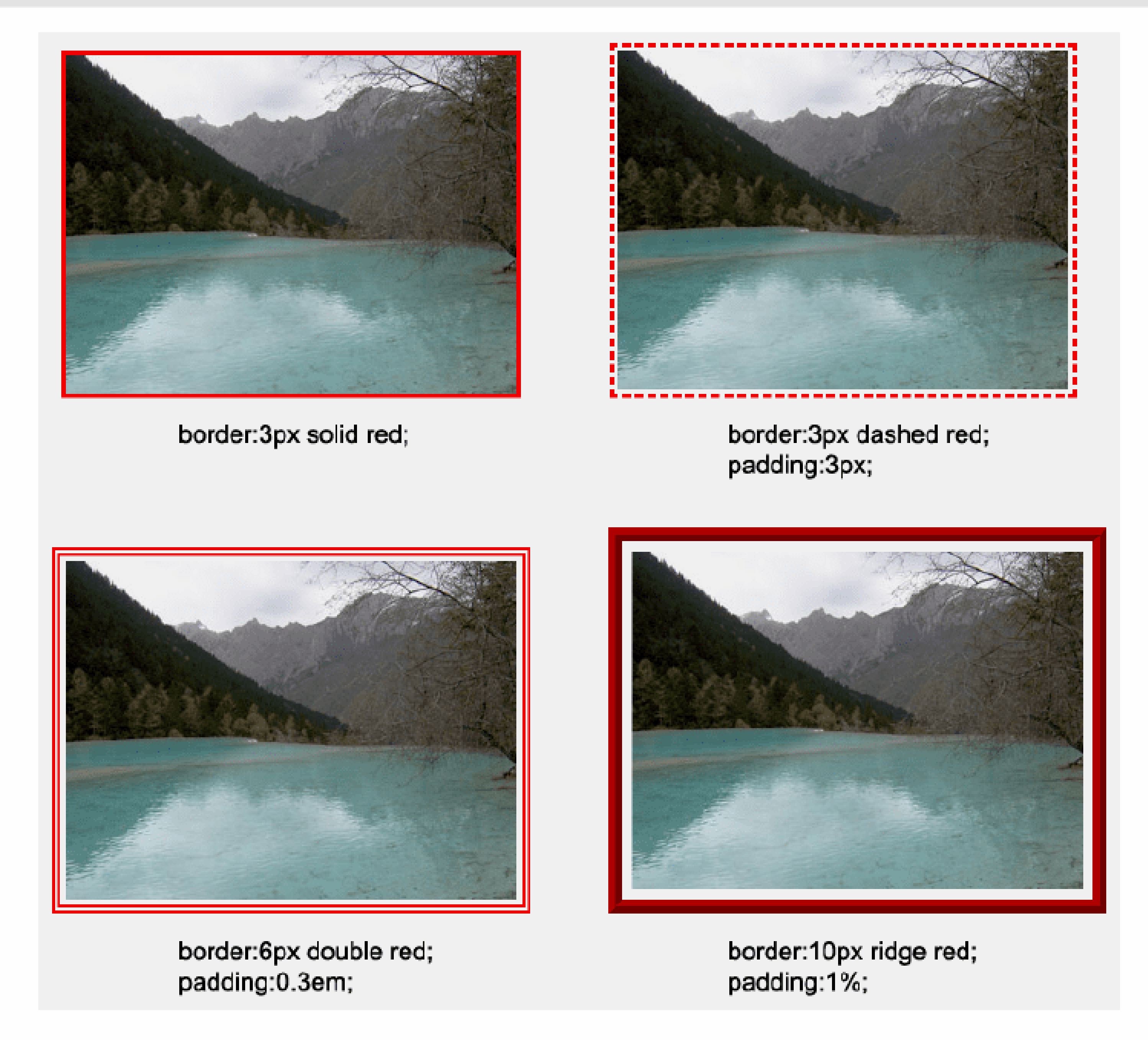


图 11-6　盒模型相关样式的运用

11.2.3　图文混排

在段落中添加图片会去除页面的单调感，让页面元素更丰富、更具吸引力。有时一条链接文字之前也会添加一些修饰性的图片，以突出细节。若想让图片和文字能更好地融合在一起，添加适当的样式信息是很有必要的。图 11-7 展示了一个图文混排的实例，可以看到图片像是嵌在了文字中，文字环绕在图片的周围。

有一家新开张的咖啡店想通过互联网宣传自己的品牌，并想在网站上介绍一些关于咖啡的知识，如果页面通篇全是文字，恐怕浏览者不会对它产生太大兴趣。如果能添加一些与咖啡相关的图片则会吸引更多的访问者。文字和图片材料已经准备好了，(X)HTML 代码如下：

```
<div id="content">
    <h1>咖啡的历史</h1>
    <p>
    <img src="images/coffee1.jpg" alt="coffee bean" width="200" height="145" />
    咖啡起源于非洲东部，这个地方就是现在的埃塞俄比亚咖法(Kaffa)省，“咖啡(coffee)”一词就是源自于省名“咖法(Kaffa)”。 传说有一个名叫 Kaldi 的放羊的牧民发现，他的羊在吃了某种植物的果实后表现得异常活泼。Kaldi 很好奇，于是亲自食用了这种果实。他发现这种果实能帮助恢复精力，这个消息马上就在当地传开了。一些僧侣在得知这种神奇的果实后，将它们凉干，带回到遥远的修道院。他们将果实泡在水中饮用，用来使自己保持清醒。</p>
    <p>咖啡果实从埃塞俄比亚传到了阿拉伯半岛，并在那里被种植。接着咖啡传到了土耳其，那里首次出现用火烘烤出来的咖啡豆。人们将烤好的咖啡豆碾碎，形成了今天咖啡的雏形。</p>
    <p>
    <img src="images/coffee2.jpg" alt="coffee cup" width="170" height="208" />
```

```
    威尼斯商人将咖啡待到了欧洲大陆。在欧洲，这种新型的饮料却遭到天主教堂严厉批评，许多人认为教皇会对咖啡下禁令，并称其为邪恶的饮料。令人们惊讶的是，教皇本人也是咖啡饮用者，并宣称为真正的基督教饮料。</p>
    <p>在 18 世纪，一个法国步兵团上尉在横跨大西洋的途中种植了一株小咖啡树。这株咖啡树被移植到了加勒比海的马提尼克岛，在随后 50 年里在该岛繁衍出一千九百万株以上咖啡树。虽然起步艰难，但咖啡种植正是借此机会传遍了中南美洲的热带地区。</p>
    <p><img src="images/coffee3.jpg" alt="coffee" width="190" height="135" />今天，咖啡成为全球的大产业，从业人员两千多万。雀巢、麦克斯韦、哥伦比亚、葛兰特、皇室哥本哈根等品牌的咖啡早已尽人皆知。这种商品在以美元为单位的全球贸易中仅次于石油居第二位。咖啡是世界上最流行的饮料，每年消耗 4000 亿杯。可以想象出，仅仅在巴西，就有五百多万人从事种植和收获，而咖啡树的数量达到 30 亿棵之巨。
    </p>
</div>
```

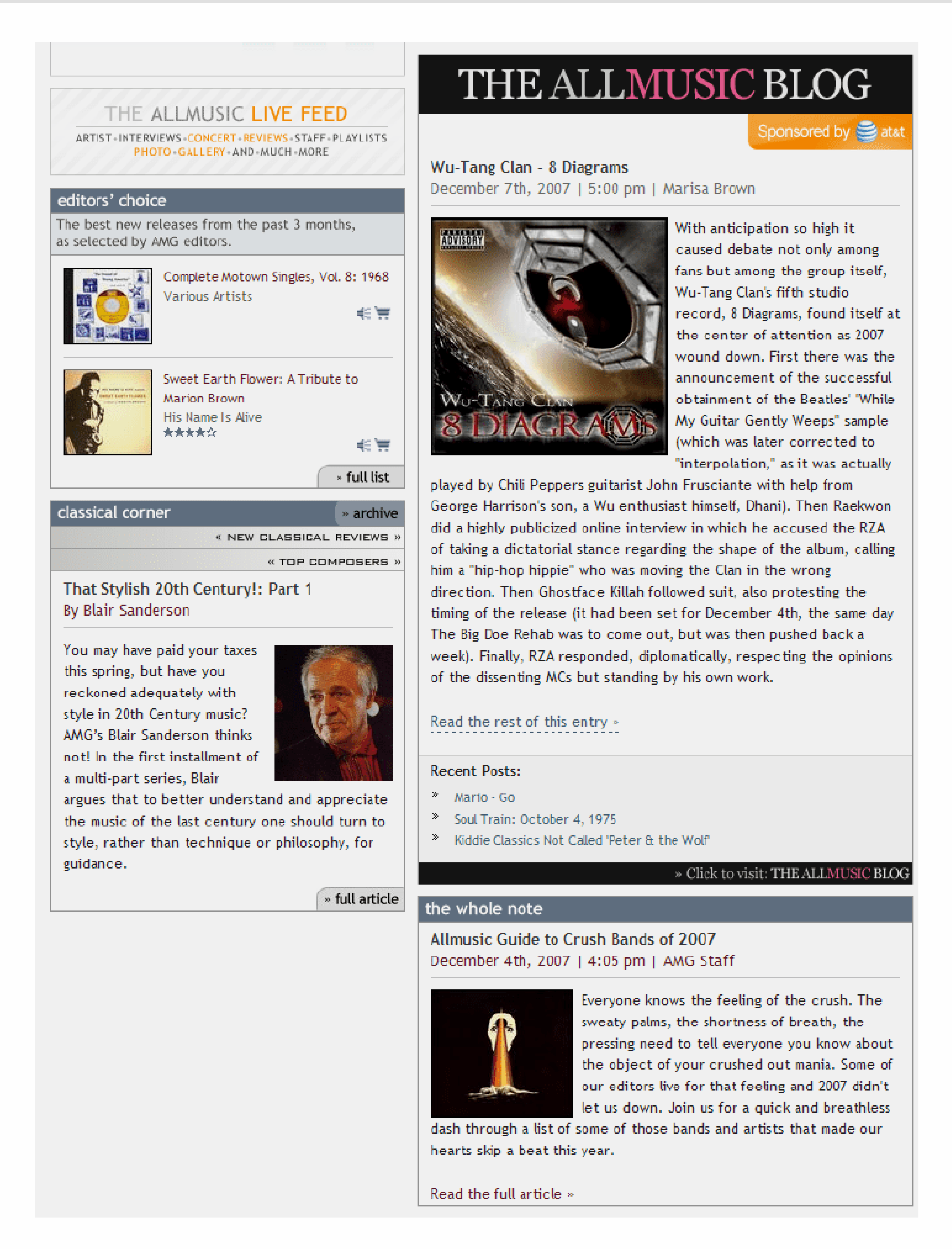


图 11-7　All Music 运用图文混排

现在没有任何样式，页面效果如图 11-8 所示。

咖啡的历史



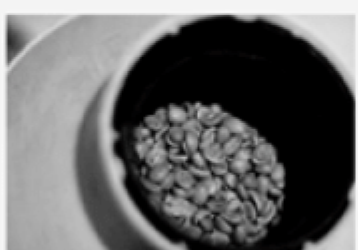
咖啡起源于非洲东部，这个地方就是现在的埃塞俄比亚咖法（Kaffa）省，“咖啡（coffee）”一词就是源自于省名“咖法（Kaffa）”。 传说有一个名叫Kaldi的放羊的牧民发现，他的羊在吃了某种植物的果实后表现得异常活泼。Kaldi很好奇，于是亲自食用了这种果实。他发现这种果实能帮助恢复精力，这个消息马上就在当地传开了。一些僧侣在得知这种神奇的果实后，将它们凉干，带回到遥远的修道院。他们将果实泡在水中饮用，用来使自己保持清醒。

咖啡果实从埃塞俄比亚传到了阿拉伯半岛，并在那里被种植。接着咖啡传到了土耳其，那里首次出现用火烘烤出来的咖啡豆。人们将烤好的咖啡豆碾碎，形成了今天咖啡的雏形。



威尼斯商人将咖啡带到了欧洲大陆。在欧洲，这种新型的饮料却遭到天主教堂严厉批评，许多人认为教皇会对咖啡下禁令，并称其为邪恶的饮料。令人们惊讶的是，教皇本人也是咖啡饮用者，并宣称为真正的基督教饮料。

在18世纪，一个法国步兵团上尉在横跨大西洋的途中种植了一株小咖啡树。这株咖啡树被移植到了加勒比海的马提尼克岛，在随后50 年里在该岛繁衍出一千九百万株以上咖啡树。虽然起步艰难，但咖啡种植正是借此机会传遍了中南美洲的热带地区。



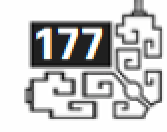
今天，咖啡成为全球的大产业，从业人员两千多万。雀巢、麦克斯韦、哥伦比亚、葛兰特、皇室哥本哈根等品牌的咖啡早已尽人皆知。这种商品在以美元为单位的全球贸易中仅次于石油居第二位。咖啡是世界上最流行的饮料，每年消耗4000 亿杯。可以想象出，仅仅在巴西，就有五百多万人从事种植和收获，而咖啡树的数量达到30亿棵之巨。

图 11-8　未添加样式之前的页面

为了使页面增强美观性，添加代码设置文字大小、行间距、文字背景颜色等，颜色选择咖啡色系，帮助突出主题。如果读者学习了前面的内容，就不难理解每条规则的含义。

代码如下：

```
body{
    color:#2D1C09;
    background-color:#734F26;
    text-align:center;
    font-family:"Times New Roman", "宋体", serif;
}
div#content{
    width:600px;
    border:1px solid #3C2811;
    background-color:#EBD8BC;
    text-align:left;
    margin-left:auto;
    margin-right:auto;
}
```



```
div#content h1{
    font-size:20px;
    text-align:center;
    padding-top:10px;
    letter-spacing:0.2em;
}
div#content p{
    font-size:14px;
    line-height:1.8em;
    text-indent:2em;
    padding:10px;
}
```

效果如图 11-9 所示。



图 11-9　增添样式后的效果

现在要给图片添加样式了，首先给 3 个 img 元素添加类名 figure，为其添加边框、填充和背景颜色，略加修饰。



修改后的 img 元素代码如下：

```
<img src="images/coffee1.jpg" class="figure" alt="coffee bean" width="200"
height="145" />
<img src="images/coffee2.jpg" class="figure" alt="coffee cup" width="170"
height="208" />
<img src="images/coffee3.jpg" class="figure" alt="coffee" width="190"
height="135" />
```

样式代码如下：

```
img.figure{
    padding:3px;
    background-color:white;
    border:1px solid #734F26;
}
```

图片的效果如图 11-10 所示。





图 11-10　添加细节内容修饰图片，右侧为局部放大图

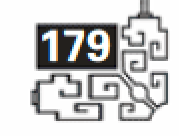
最后一步就是要实现文字环绕在图片周围的效果了，并且我们想让第二幅图片向右侧对齐，这些效果都要依靠于 float 属性，它是使用 CSS 进行页面布局的关键之一，我们在第四篇会进行详细的讲解。简单地说，float 属性可改变浏览器对于(X)HTML 文档的显示方式。默认情况下，(X)HTML 文档中的元素会按照顺序由上至下出现在浏览器窗口中，内联元素会从左向右显示，块级元素会独占一行。如图 11-9 所示，标题、段落由上至下依次显示，图片是内联级的元素，与文字从左向右依次显示。

float 属性可让元素向左或向右浮动，使它脱离原有的排列方式，挨着浮动元素的内容会把该元素包围起来，因此使用 float 即可轻松实现文字环绕图片的效果。

我们添加两个规则：

```
.leftAlign{
    float:left;
}
.rightAlign{
    float:right;
}
```

并相应地给 img 元素增加类名：



```
<img src="images/coffee1.jpg" class="figure leftAlign" alt="coffee bean"
width="200" height="145" />
<img src="images/coffee2.jpg" class="figure rightAlign" alt="coffee cup"
width="170" height="208" />
<img src="images/coffee3.jpg" class="figure leftAlign" alt="coffee"
width="190" height="135" />
```

效果如图 11-11 所示。



图 11-11　float 属性实现文字环绕图片

文字已经环绕在图片周围了，但是细节处理得不是很好，注意向左浮动的两幅图片右侧和文字之间没有空隙，不是很美观，我们给 img 添加 id 属性，并设置一定的边距。

(X)HTML 代码如下：

```
<img src="images/coffee1.jpg" class="figure leftAlign" id="figureA"
alt="coffee bean" width="200" height="145" />
<img src="images/coffee2.jpg" class="figure rightAlign" id="figureB"
alt="coffee cup" width="170" height="208" />
<img src="images/coffee3.jpg" class="figure leftAlign" id="figureC"
alt="coffee" width="190" height="135" />
```

CSS 代码如下：



```
img#figureA, img#figureC{
    margin:0 10px 0 0;
}
img#figureB{
    margin:0 0 0 10px;
}
```

为了统一边距大小，我们给第二幅图片也设置了同样的边距，只不过边距的方向不同。最终效果如图 11-12 所示。

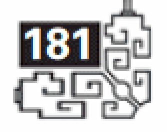


图 11-12　图文混排最终效果

除了用在正文中，图片还经常被用来装饰一些链接或标题文字(见图 11-13)。



图 11-13　图像也经常用来装饰链接和标题



通常图片可以单独放在<span>或<div>标记中，或者利用背景图片属性实现，这里介绍的方法是直接控制 img 元素样式而不需要任何辅助标记。

现有如下(X)HTML 和 CSS 代码：

```
a{
    font-size:12px;
    text-decoration:none;
}
a:link, a:visited{
    color:#1750A5;
}
a:hover{
    color:#FC7614;
}

<a href="http://www.huistd.com">欢迎访问工作室网站</a>
```

我们在 a 元素内文字之前插入一张图片作为装饰，代码如下：

```
<a href="http://www.huistd.com"><img src="images/link.gif" alt="link" />欢迎
访问工作室网站</a>
```

插入后效果如图 11-14 所示，看起来这种效果不能令人满意。

图 11-14　插入图像后的效果

添加如下代码去掉图像边框并让图片和右侧文字保持一定距离：

```
a img{
    border:0;
    margin-right:5px;
}
```

如图 11-15 所示为修改后的效果。

图 11-15　修改后效果

现在感觉图像在竖直方向的位置不够理想，可以使用 margin-bottom 属性为图像多添加一些下边距，另外，属性 vertical-align 也能起到类似的作用，但是对于同样的参数各个浏览器显示却不一致，因此不推荐使用，我们添加 1px 的下边距，完整样式如下：

```
a{
    font-size:12px;
    text-decoration:none;
}
a:link, a:visited{
    color:#1750A5;
```

```
}
a:hover{
    color:#FC7614;
}
a img{
    border:0;
    margin-right:5px;
    margin-bottom:1px;
}
```

图 11-16 为最终效果。

欢迎访问工作室网站

图 11-16　为链接添加装饰图片的最终效果

11.3　背景和图像

上面介绍的只是利用一些公共属性来为插入在页面中的图片添加样式，现在该轮到 CSS 本身发挥其添加图像的功能了。background-image 属性是 CSS 添加图像的关键，添加的图像是作为元素的背景存在的，再配合其他与背景相关的属性就能完成样式控制等功能。如图 11-17 所示为 Tennessee 旅游发展部门(http://tnvacation.com/)的网站，该站点就是利用 CSS 的背景属性添加了丰富的背景图像。

在背景相关属性之中只有一个是和图像无关的，但它也非常重要，这就是背景颜色属性 background-color，用于设置元素的背景颜色，默认值为 transparent，即为透明。该属性的使用频率相当高，很多样式效果都是依靠这个属性来完成的。

其余背景相关属性为 background-repeat、background-attachment 和 background-position，它们都与背景图像相关，通常配合在一起使用，下面我们就分别进行介绍。

11.3.1　设置背景图像

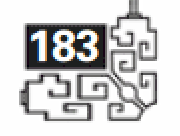
为页面元素添加一张背景图像很简单，只需一条声明，请看下面的示例。

CSS 和(X)HTML 代码：

```
div#sideBar{
    width:300px;
    height:300px;
    background-image:url(images/flowers.jpg);
}

<div id="sideBar">div 中文字</div>
```

规则指定了 div 的宽度和高度，从图 11-18 可以看出背景图像以平铺的方式显示在元素中，



默认情况下，背景图像在水平和垂直方向上都是重复出现的。有时候需要利用这种特性来实现某些效果，对于一些运用在整个页面的背景图片，该方法使用率非常高。

图 11-17　Tennessee 旅游开发部门网站

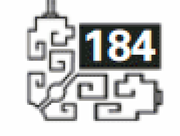




图 11-18　给 div 元素添加背景

在使用 background-image 属性时需要注意，url 中的图片路径一定是相对于 CSS 出现的位置的。假如上例中的 CSS 样式是由外部导入的，且样式文件与该(X)HTML 文件位于不同的目录层次中，那么图片路径就要相对于该 CSS 文件进行变化。其他含有路径的样式属性也是如此，资源位置是相对于 CSS 文件的。

11.3.2　平铺背景图像

background-repeat 属性可以控制背景图像平铺的方式，其取值和含义如下所示。

- repeat：背景图像在纵向和横向上平铺。
- no-repeat：背景图像不平铺。
- repeat-x：背景图像横向平铺。
- repeat-y：背景图像纵向平铺。

例如，取消上例中背景图像的平铺：

```
background-repeat:no-repeat;
background-color:#CCCCCC;
```

为了更好地显示效果，将 div 元素背景设为灰色。从图 11-19 可看出，在 div 元素内图像只出现了一次，没有任何平铺效果。

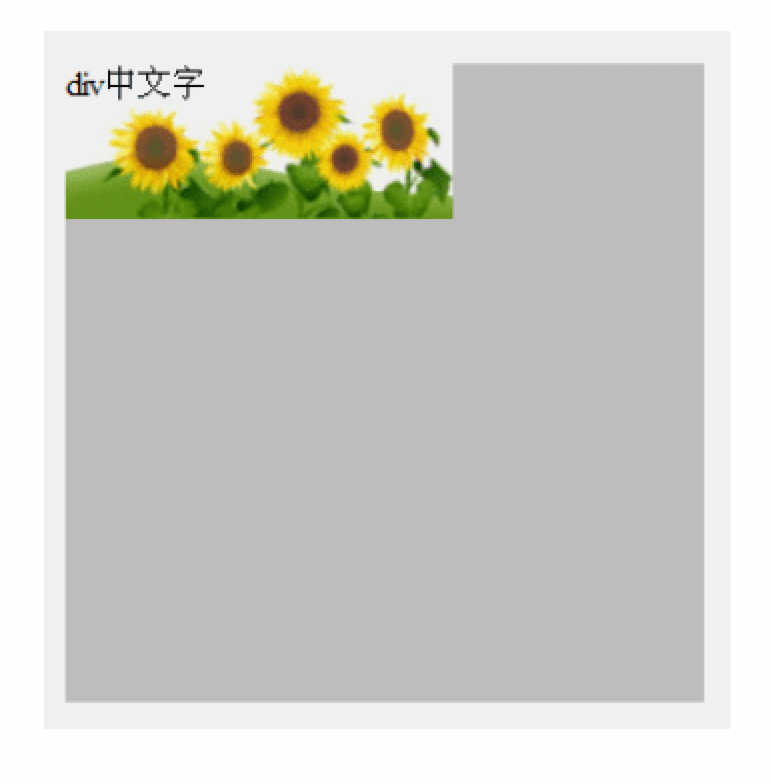
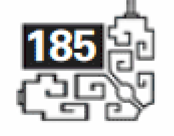


图 11-19　取消平铺



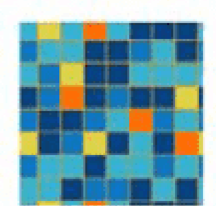
一些网站利用背景的不同平铺方式来实现渐变或纹理效果，例如微软中国网站首页的顶部背景就是水平平铺一幅 1px 宽 378px 高的图片实现的(见图 11-20)。由于图像尺寸很小，以这种方式制作背景渐变效果可大大节省网络带宽，加快页面下载的速度。

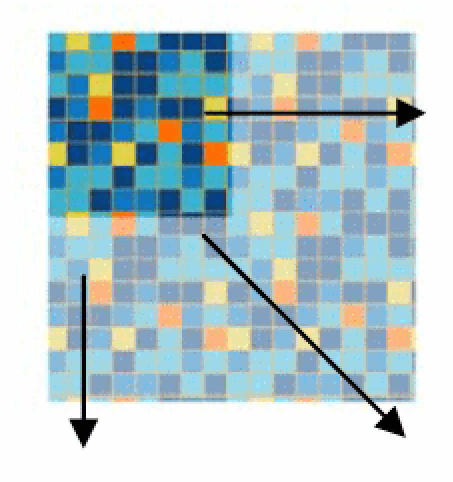
图 11-20　微软中国网站首页的背景图片使用水平方式平铺而成

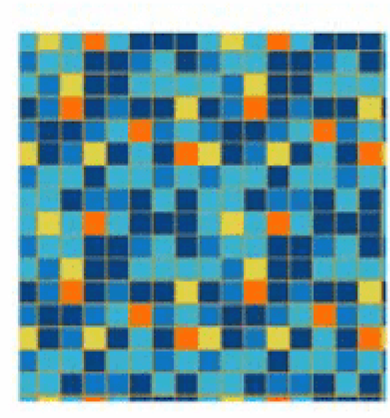
有些网站的背景是在水平和垂直两个方向上平铺图片实现的，另有一些网站，背景看上去像是一些毫无规律的纹理，实际上是平铺无缝贴图实现的。所谓无缝贴图，就是指图片左边和右边、顶部和底部连接在一起显示时，看不出丝毫缝隙，因此可以使用较小的图片制作整个页面背景。

例如 Groninger Museum(http://www.groningermuseum.nl/)网站页面背景的马赛克就是使用一张图像平铺而成的，图片上下边、左边可以完美地拼接在一起(见图 11-21)。









背景图片　　　　横向和纵向平铺构成整个背景

图 11-21　使用小图平铺构成背景

网络上有相当丰富的无缝贴图资源，都可以直接拿来或稍加修改后应用到网页背景中，如图 11-22 所示就是下载并经过简单处理的一张砖墙图像，利用它可以实现砖墙背景的效果。

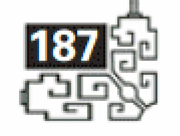


图 11-22　砖墙素材

在页面中添加如下 CSS 代码，指定背景图片横向纵向平铺。

```
body{
    background-image:url(images/backwall.jpg);
    background-repeat:repeat;
}
```

结果如图 11-23 所示，图片可以很自然地拼接在一起，形成整个背景。



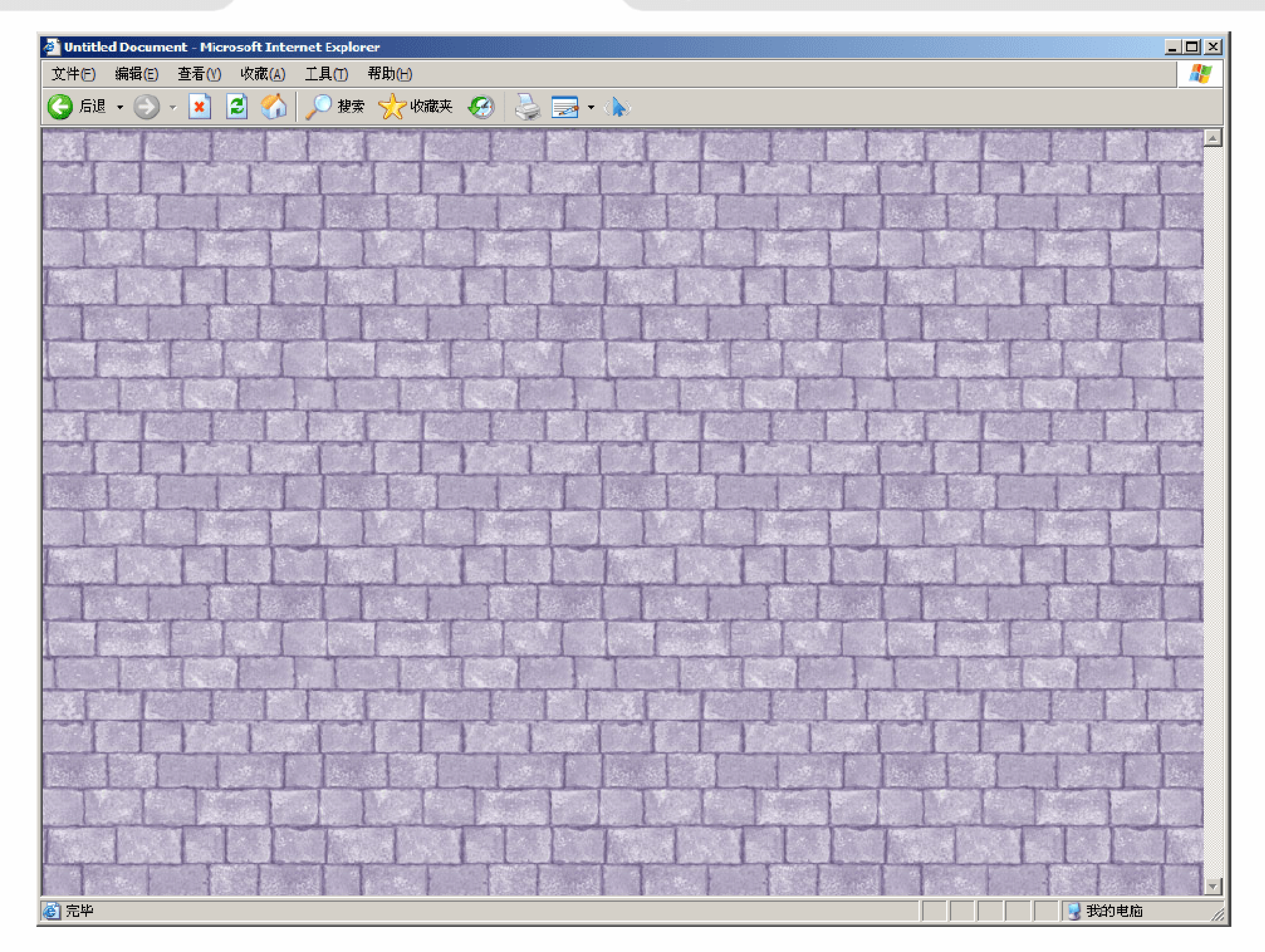


图 11-23　平铺图像制作页面背景

11.3.3　背景图像位置

通过控制图像的平铺方式可以制作出不同样式的背景。CSS 提供了 background-position 属性，可用来控制背景图像的位置，结合平铺属性一起使用可精确指定图像从哪里开始进行平铺。

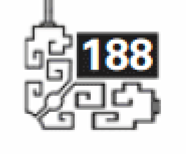
background-position 属性值为两个，前者表示水平位置后者表示垂直位置。它的属性值可以是关键字、长度值和百分比。

1. 关键字

关键字包括 left、right、center、top、bottom，水平和垂直方向组合起来使用可组合出 9 种不同的位置(图 11-24)。

微软中国有限公司的首页背景顶部和底部分别有两个渐变，分别在两个 div 元素中添加了背景图，如果查看代码即可发现它们的 background-position 属性分别为 top 和 bottom。制作渐变背景图的一般步骤为，先在图像处理软件中设计好背景效果图，利用切片工具提取 1px 宽的图像并保存成适当的格式，最后添加 CSS 样式到页面中。

现在我们就模仿微软首页制作顶部和底部均包含渐变的效果，首先使用图像处理软件设计好效果图(见图 11-25)。把顶部和底部渐变所需图像切割出来，分别保存为 bgTop.gif 和 bgBottom.gif，它们尺寸为 1px 宽 200px 高。



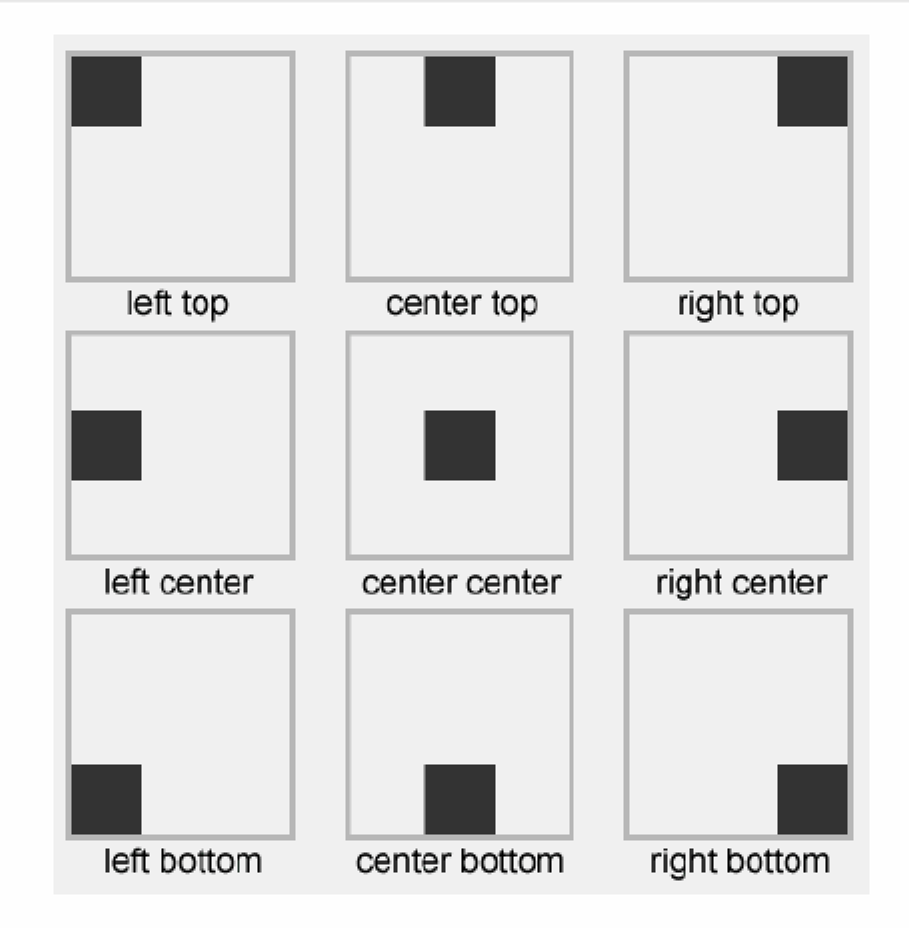


图 11-24　background-position 属性示意图

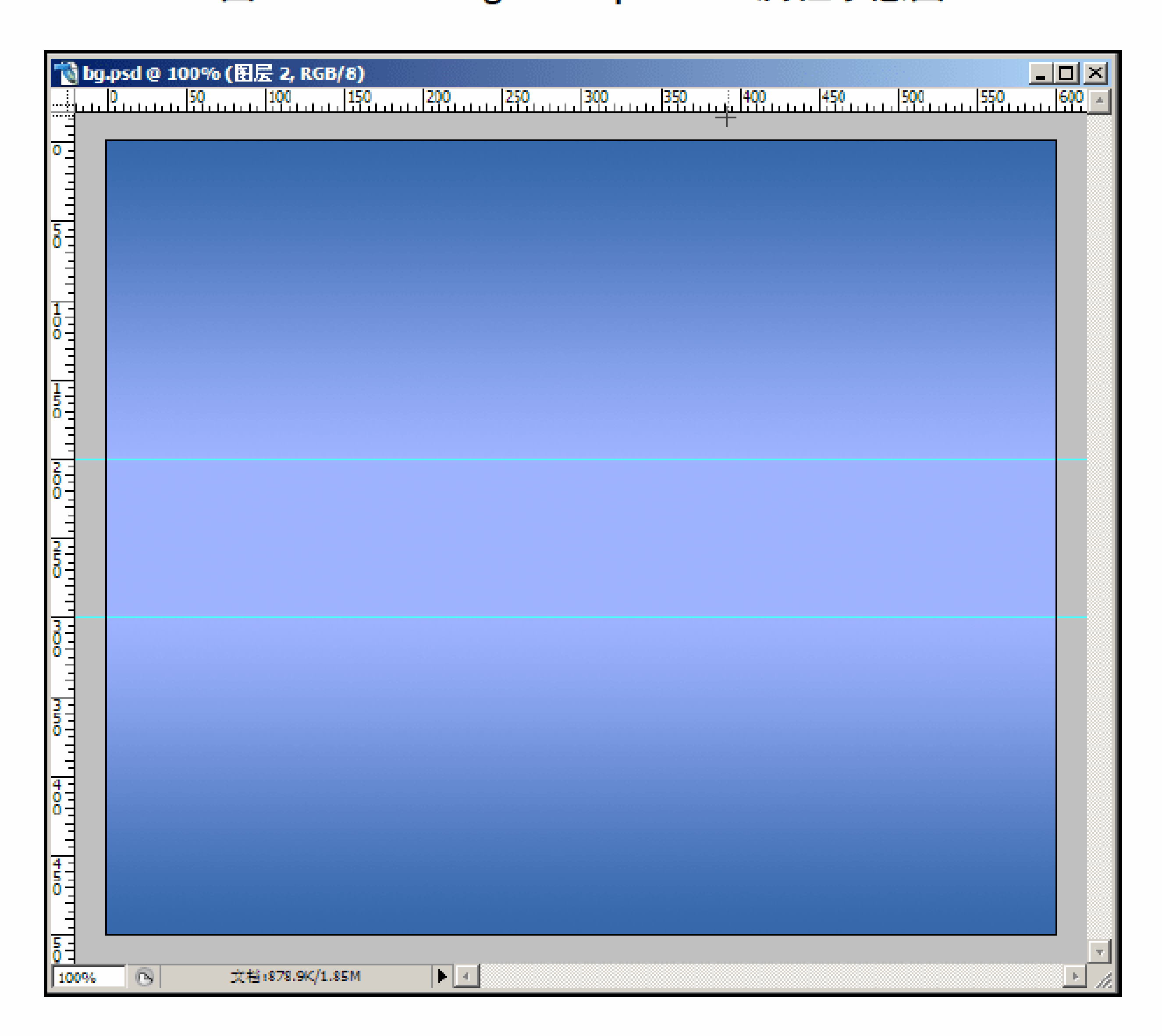
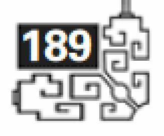


图 11-25　设计好渐变效果图

代码如下所示：

```
*{
    margin:0;
    padding:0;
}
body{
    background-image:url(images/bgBottom.gif);
    background-repeat:repeat-x;
    background-position:left bottom;
    background-color:#9EB2FF;
}
```

```
div#container{
    background-image:url(images/bgTop.gif);
    background-repeat:repeat-x;
    background-position:left top;
}
div#content{
    height:500px;
}

<div id="container">
    <div id="content"></div>
</div>
```

规则首先设定所有元素的边距和填充为 0，然后在 body 元素中添加底部的背景图，注意到其中的 background-position 属性值为“left bottom”，图像将在 body 元素左下侧开始沿着 x 轴方向(即水平方向)进行平铺。另创建一个 div 元素放置顶部的背景图，背景图从该元素左上侧开始进行水平平铺。最后再添加一个 div 元素，设定其高度为 500px，用来模拟页面中的内容，让外层的#container 能够有足够的高度把背景图显示完整。图 11-26 为最终效果。

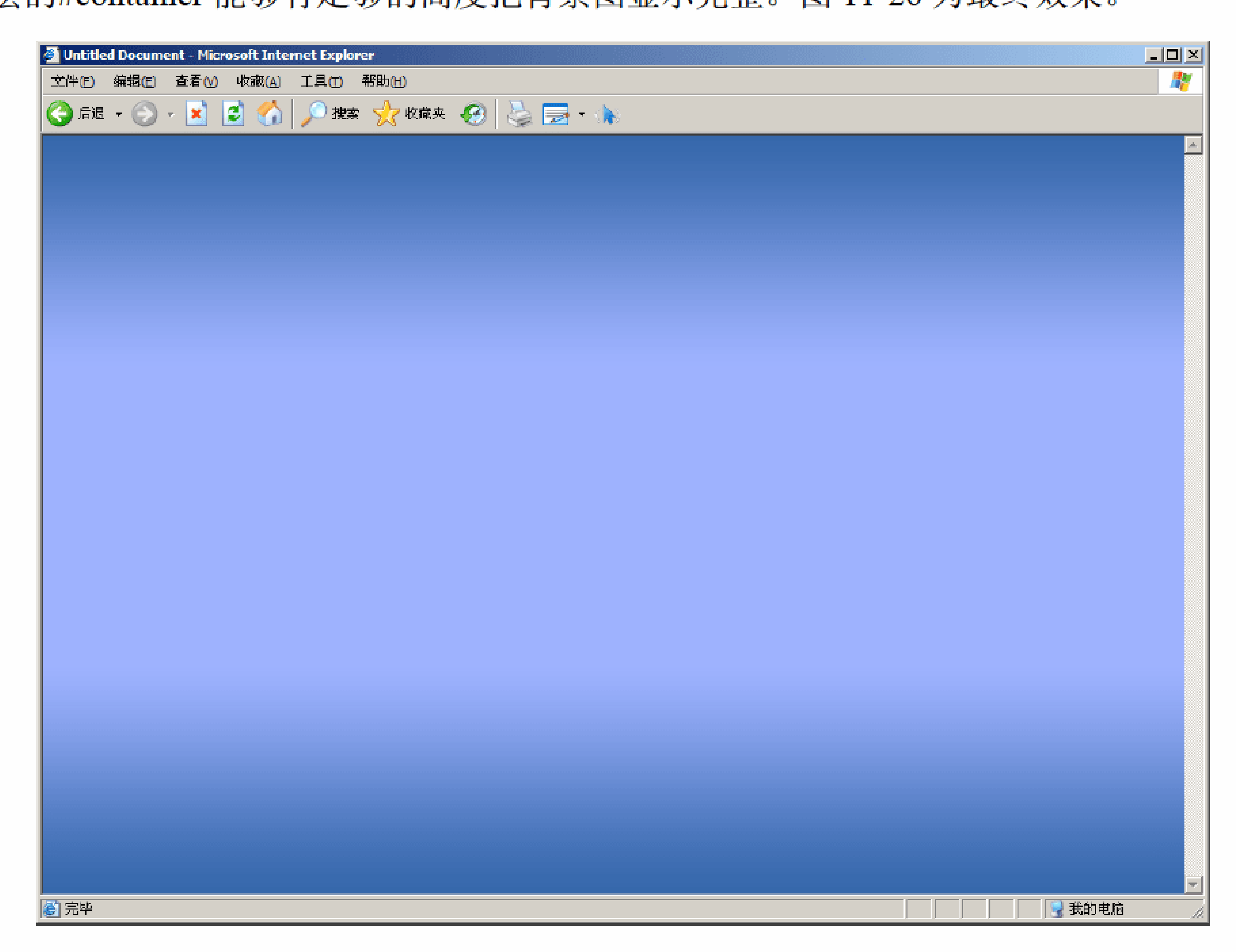
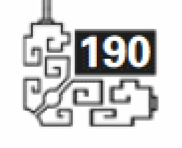


图 11-26　设定背景图

2. 长度值

background-position 的属性值还可以是长度值，其中单位 px 和 em 比较常用，这时属性第一个值表示图像左边距父元素左边的距离，第二个值表示图像顶端距父元素顶端的距离，且该属性值可取负数。如图 11-27 所示为两个属性值的含义。



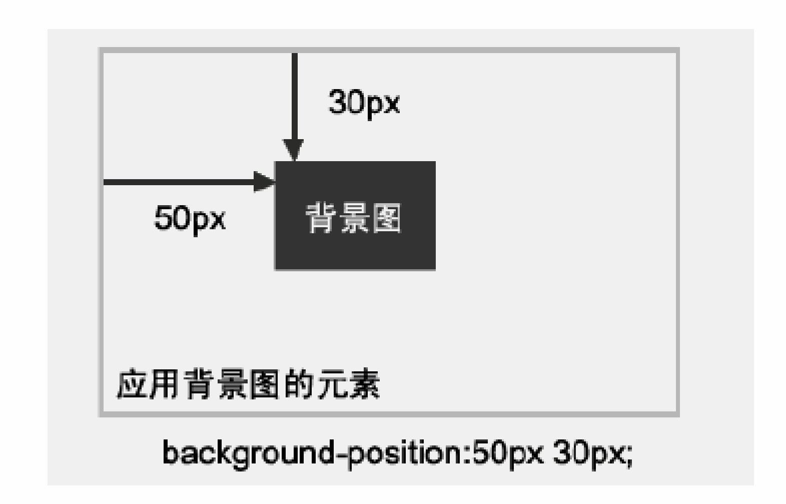


图 11-27　长度值的含义

还记得 11.2 节最后讲的如何添加装饰图片的内容吗？我们说过，除了直接插入图片外还可以利用 CSS 背景图像属性来完成。首先去掉(X)HTML 代码中<img>标记：

```
<a href="http://www.huistd.com">欢迎访问工作室网站</a>
```

再添加如下 CSS 样式：

```
a{
    font-size:12px;
    text-decoration:none;
    background-image:url(images/link.gif);
    background-repeat:no-repeat;
}
a:link, a:visited{
    color:#1750A5;
}
a:hover{
    color:#FC7614;
}
```

在图 11-28 中，我们发现图像被盖在文字下面了，这并不是我们想要的结果，需要给文字添加一些左填充，使其向右移动，让图像露出来。

欢迎访问工作室网站

图 11-28　给链接添加背景图片

在选择符 a 中添加如下声明，让文字向右移动一段距离。

```
padding-left:10px;
```

padding 属性只会影响盒模型中内容部分，背景图片是不会移动的，这时的效果如图 11-29 所示。

欢迎访问工作室网站

图 11-29　右移文字露出图片

最后就要调整背景图片位置了，在选择符 a 中添加如下规则：

```
background-position:0 2px;
```

该规则让图像向下移动 2 个像素，图 11-30 为最终效果。

欢迎访问工作室网站

图 11-30　使用背景图像添加装饰

3. 百分比

百分比数值的含义比较有意思，首先根据百分比值在背景图像上确定一个位置作为基准点，然后用该位置去对齐父元素中百分比值所设置的位置，图 11-31 中处于父元素 25%和 75%的背景图像就很能说明这个特点。

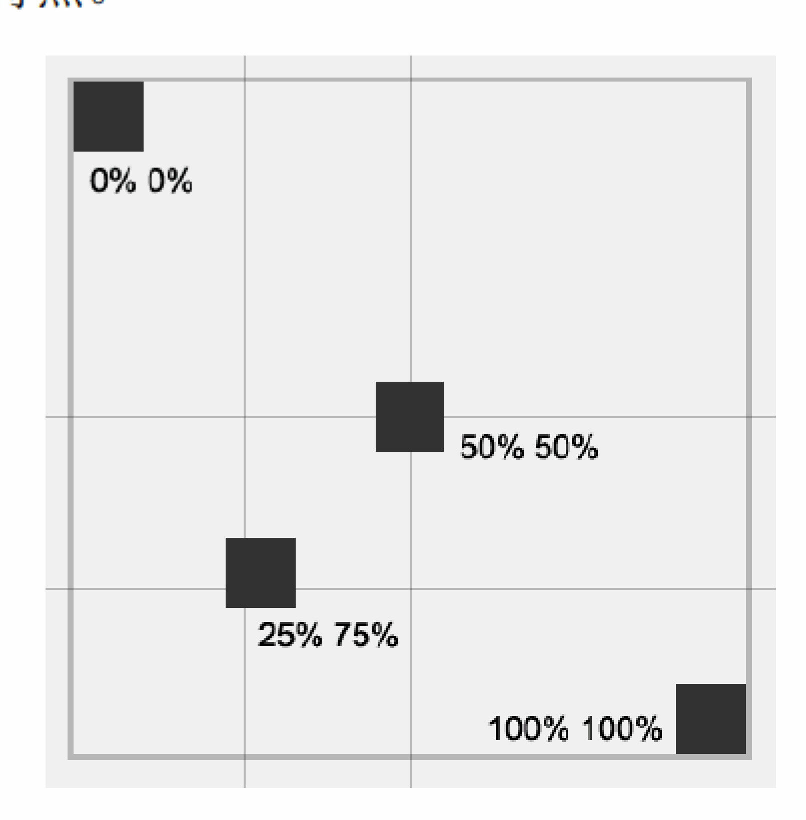


图 11-31　百分比值的含义

当页面元素需要动态地按比例变化的时候，百分比就会派上用场。百分比、长度值和关键字可以混合在一起使用。

长度值和百分比可以带符号，使用负值可达到只显示部分图片的效果。比如下面的代码就可以让砖墙图片只露出一小部分：

```
div{
    width:400px;
    height:200px;
    background-image:url(images/backwall.jpg);
    background-repeat:no-repeat;
    background-position:-120px -120px;
    background-color:yellow;
}

<div></div>
```

代码效果如图 11-32 所示。

图 11-33 展示了 background-position 属性使用负数值的含义。

与位置相关的 CSS 属性还有一个，即 background-attachment，该属性有两个值，即 fixed 和 scroll。fixed 的含义是将背景图像固定住，拖动滚动条也不会影响背景图的位置。而 scroll 的含义是当拖动滚动条时，背景图像会跟随页面其他内容一起滚动，这也是浏览器默认的方式。W3C 的 CSS 规范页面就使用了固定的背景图(见图 11-34)。

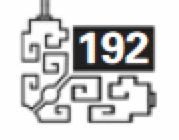




图 11-32　使用负值定位背景图片

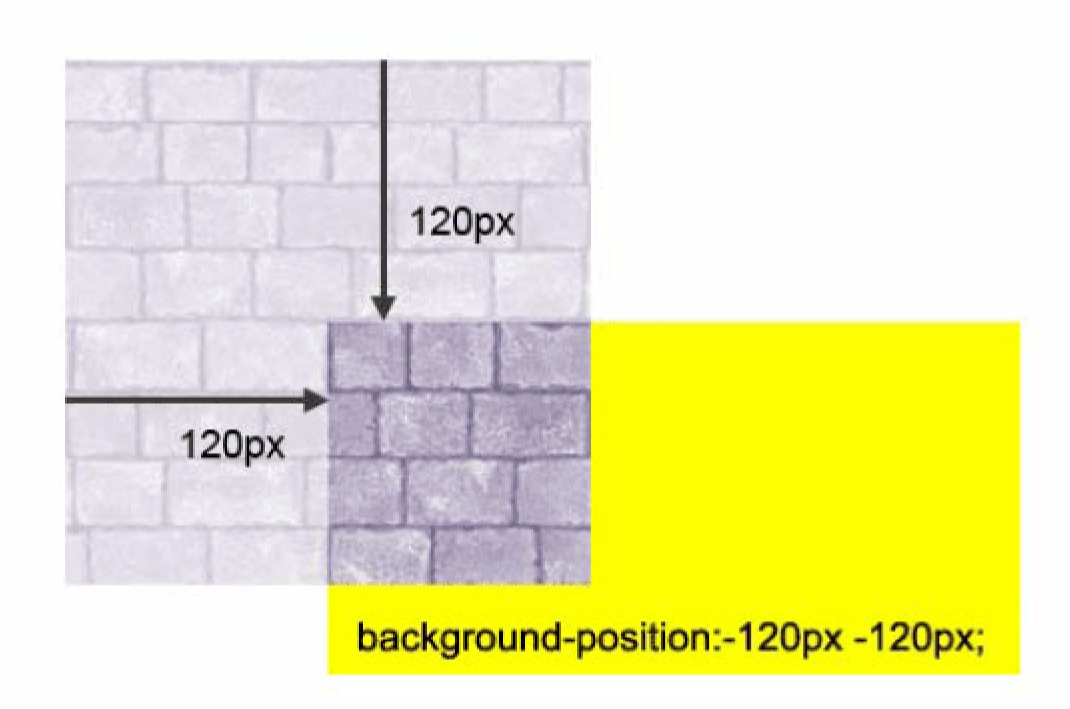


图 11-33　background-position 使用负值的含义

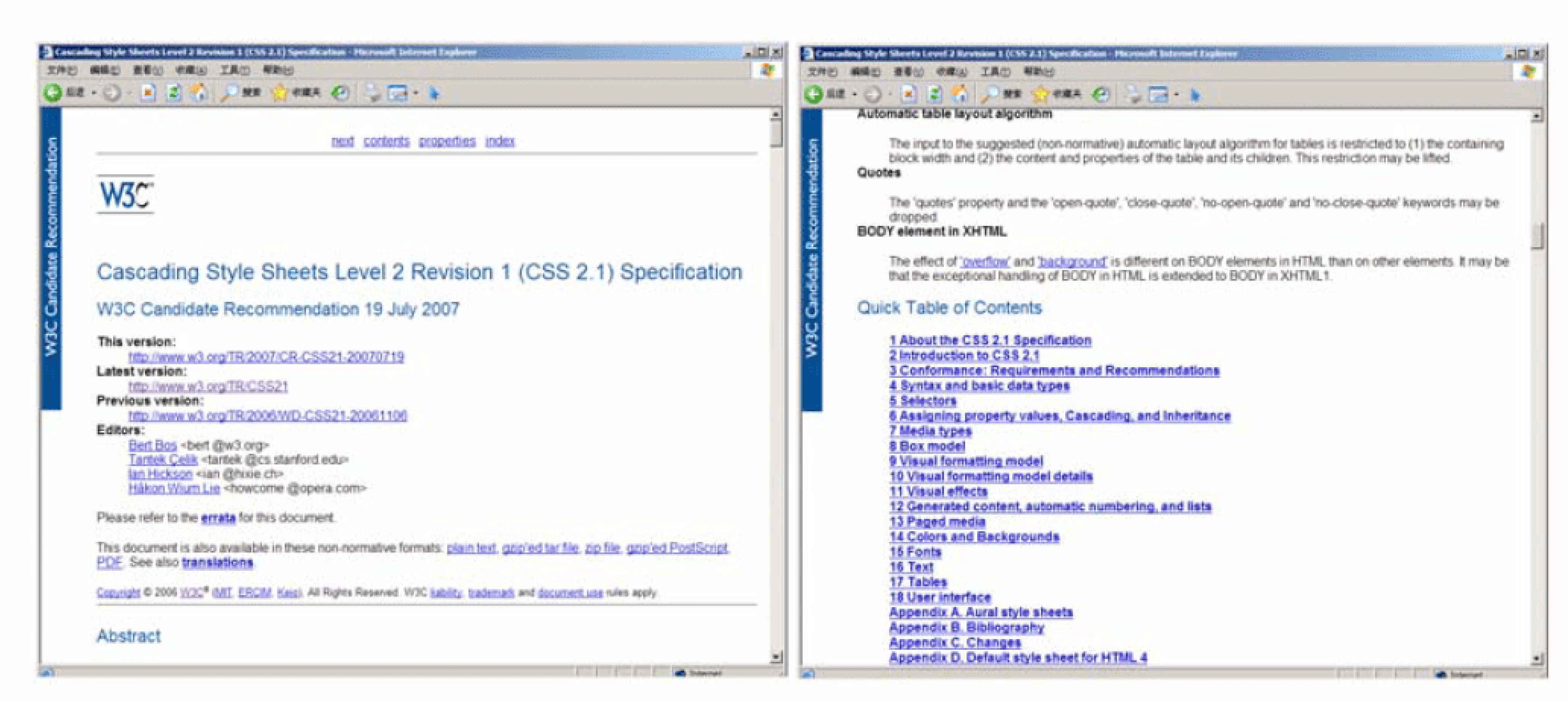
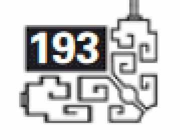


图 11-34　拖动浏览器滚动条，页面左侧蓝色背景图片位置始终不变

注意：在 IE6 中，如果 background-attachment 属性应用在 body 以外的元素，则 fixed 属性值将不起作用，图像依然随着滚动条移动。

11.3.4　background 属性

经过了前面的学习，读者已经掌握了各种控制背景图像的属性。与 font 属性类似，CSS 提供了 background 属性作为各种背景相关属性的简写属性。属性值的顺序可任意。以下几条规则



的效果是等同的：

```
div{
    background-color:white;
    background-image:url(images/bg.gif);
    background-position:top left;
    background-repeat:repeat-x;
    background-attachment:fixed;
}
div{
    background:white url(images/bg.gif) top left repeat-x fixed;
}
div{
    background:url(images/bg.gif) repeat-x top left fixed white;
}
```

尽管各个属性出现的顺序可任意，但是表示位置的两个属性值不能分开，并要保证前者表示水平方向，后者表示垂直方向。

11.4 图 像 实 战

11.4.1 网络相册

网络相册可算是 Web 2.0 中最为常见的应用了，用户可以将自己的图像、照片上传至网络相册，与他人分享。和当初许多人使用表格实现布局一样，图像的排版也经常通过表格来完成，本实战将向读者介绍另一种方法，它的优势在于使用简洁的(X)HTML 代码放置图像，其余的事情都交给 CSS 来完成。

首先我们来看看(X)HTML 的结构：

```
<div class="photobox">
   <div class="photo">
   <img src="images/figure1.jpg" alt="九寨沟" width="200" height="150" /></div>
   <p>九寨沟长河</p>
</div>
```

图像被放置在一个 div 中，并和放置说明文字的 p 元素放置在另一个 div 元素中。注意，img 元素的结束位置与后面的</div>标记之间不要有任何空白存在，后面会讲为什么要这样做。按照这个基本结构添加更多的图像，代码如下(其中省略了一些重复代码)：

```
<div id="container">
   <h1>我的相册</h1>
   <div class="photobox">
      <div class="photo">
<img src="images/figure1.jpg" alt="九寨沟" width="200" height="150" /></div>
      <p>九寨沟长河</p>
      </div>
```



```
    <div class="photobox">
        <div class="photo">
<img src="images/figure2.jpg" alt="樱花" width="200" height="150" /></div>
        <p>樱花</p>
    </div>
    ...
    <div class="photobox">
        <div class="photo">
<img src="images/figure6.jpg" alt="电脑" width="200" height="150" /></div>
        <p>我的电脑</p>
    </div>
</div>
```

我们添加 6 幅图片到代码中，最后增加一个标题，并把这些元素都放到#container 容器中。(X)HTML 准备完毕，现在开始给 img 和 p 元素添加样式代码：

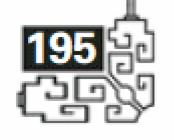
```
.photo img{
    margin:0;
    padding:10px;
    background:#FFFBCB;
    border:1px dashed #D8D399;
}
.photobox p{
    font-size:12px;
    margin:0;
    padding:3px 0 3px 24px;
    background:#FFFDE6 url(images/note.gif) 5px 4px no-repeat;
}
```

以上两条 CSS 规则给 img 元素添加了填充、背景色和边框样式，设置 p 元素的字体大小、边距填充和背景色和图片。打开浏览器观察一下效果(见图 11-35)。



图 11-35　给 img 和 p 添加样式

图 11-36 对比了 IE 和 Firefox 的显示效果，可以发现 p 元素中的上填充在两个浏览器中的显示效果不一致。



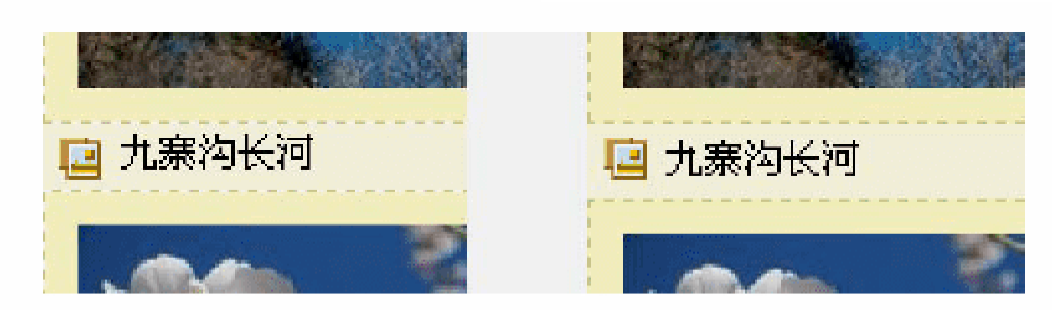


图 11-36　IE(左)和 Firefox(右)显示效果对比，它们的上填充大小不一致

为此我们通过 IE 条件注释添加一些 IE 特定的样式：

```
<!--[if IE]>
<style type="text/css">
.photobox p{
    padding-top:5px;
}
</style>
<![endif]-->
```

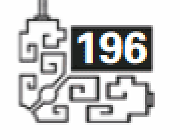
现在每一个图像由上至下依次排列，这是 div 元素的默认排列方式，而我们希望图像能水平排列，于是添加如下样式：

```
.photobox{
    float:left;
    width:222px;                /* 200 + 20 + 2 = 222px */
    margin:0 10px 10px 10px;
    display:inline;
}
```

这里又利用了 float 属性，.photobox 将向左浮动，水平方向排列，当水平空间不足时自动转到下一行。元素宽度设为 222px，它是图像宽度(200px)、图像左右填充(各 10px，共 20px)和左右边框(各 1px，共 2px)大小之和。margin 属性的作用在于使每个图像之间存在一定空隙。display 属性的作用在第 17 章浮动部分会详细介绍，它是用来解决 IE 浮动元素边距加倍的问题。预览一下现在的页面效果，如图 11-37 所示。



图 11-37　添加浮动属性使图像水平排列(此效果可能会由于浏览器窗口的大小不同而不同)



如果调整浏览器窗口的大小，我们会发现图片的位置会发生变化(见图 11-38)。这是由浮动属性产生的。



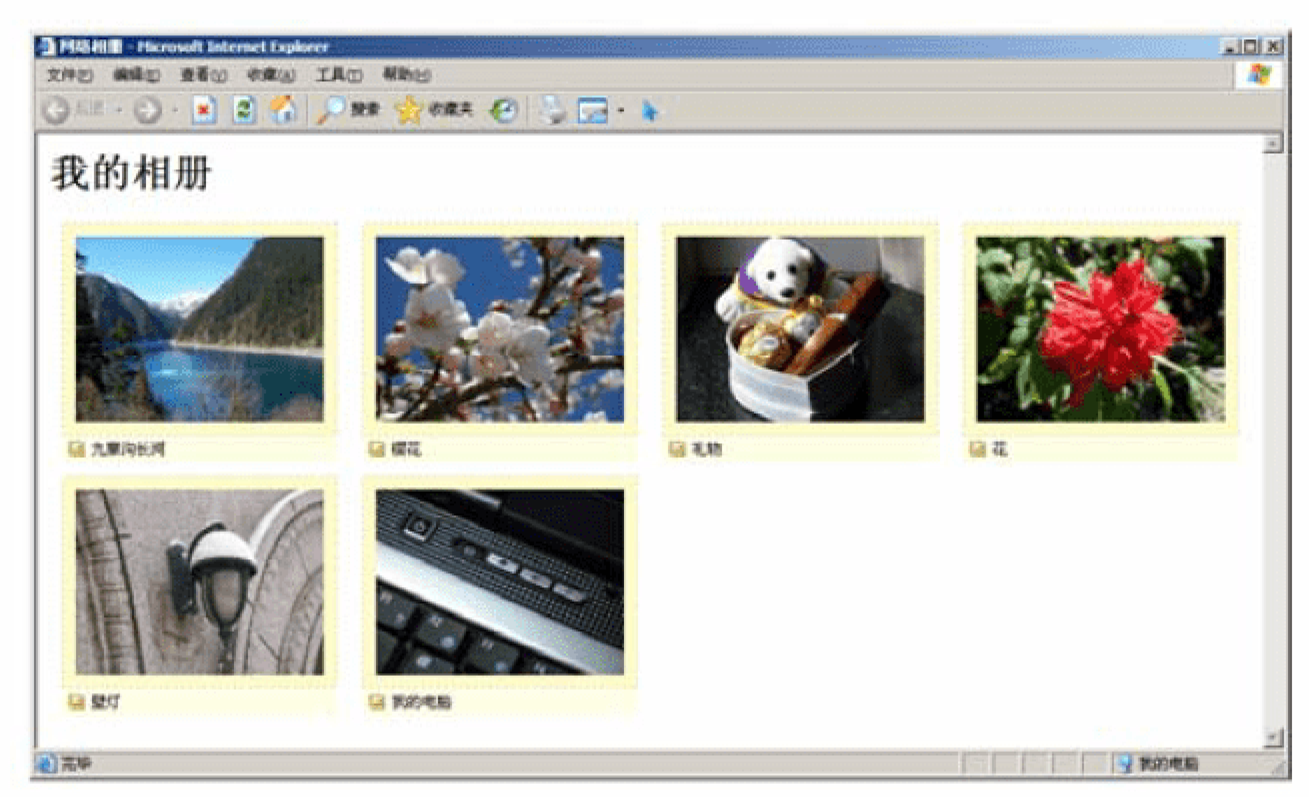


图 11-38　图片位置会随着浏览器窗口宽度进行变化

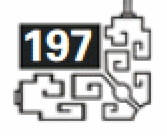
我们希望页面的布局在任何情况下都是统一的，每行固定显示三张图片。这可以通过给最外侧的容器#container 添加固定的宽度来完成：

```
#container{
    width:730px;
    margin:0 auto;
}
```

每个.photobox 的宽度为 222px，再加上左右边距一共是 242px，一行三幅图片总共需要 726px，所以#container 的宽度定为整数 730px。margin 属性使#container 居中对齐。

最后我们给标题 h1 元素添加一些样式，本实例的完整 CSS 代码如下：

```
<style type="text/css">
.photo img{
    margin:0;
    padding:10px;
    background:#FFFBCB;
    border:1px dashed #D8D399;
}
.photobox p{
    font-size:12px;
    margin:0;
    padding:3px 0 3px 24px;
    background:#FFFDE6 url(images/note.gif) 5px 4px no-repeat;
}
.photobox{
    float:left;
    width:222px;                /* 200 + 20 + 2 = 222px */
    margin:0 10px 10px 10px;
    display:inline;
}
```



```
#container{
    width:730px;
    margin:0 auto;
}

h1{
    font-size:24px;
    padding:3px 0 10px 60px;
    border-bottom:2px solid #DDD;
    background:url(images/photologo.gif) 5px 2px no-repeat;
}
</style>
<!--[if IE]>
<style type="text/css">
.photobox p{
    padding-top:5px;
}
</style>
<![endif]-->
```

页面的最终效果如图 11-39 所示。

在给出本实例的(X)HTML 代码的时候，我们说过 img 元素和后面的标记之间不能留有任何空白，那么如果有空白存在会如何呢，现在我们就来试验一下。把第一幅图像对应的 img 元素后面加一些空白(空格、换行或者制表符都可以)，用 IE 浏览器一下页面(见图 11-40)。你会发现第一幅图像和下面的区域之间增加了一些空间，且导致第二行水平空间不足，使得最后一幅图像被挤到第三行了。



图 11-39　网络相册的最终效果

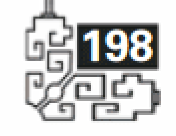




图 11-40　增加空白之后的页面效果(IE 浏览器)

这个多余的空间就是由 img 元素和其后的标记之间的空白产生的，这是 IE 浏览器的一个问题，Firefox 和 Opera 则会正常显示。由于该问题是由空白导致的，所以也可称其为 IE 空白问题(IE Whitespace Bug)。该问题不仅出现在 img 元素中，在其他情况下也有可能发生，读者在开发过程中需要注意。

11.4.2　圆角设计

圆角样式在 Web 设计中经常出现，往往一个区域、一块内容都会采用圆角设计方案，使页面看起来更美观(见图 11-41)。



图 11-41　Komodo Media 网站的页面使用了圆角效果(http://www.komodomedia.com)

首先准备好 4 个圆角所使用的图片(见图 11-42)，它们各用于矩形区域的 4 个顶点。



图 11-42　圆角图片

我们将一个段落设计成圆角样式，(X)HTML 代码如下：

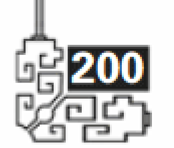
```
<div>
    <h3>&bull; 圆角样式 &bull;</h3>
    <p>
    圆角样式经常用于网站设计当中，使用它可以提高页面的美观性，使页面元素看上去更柔和。
    </p>
</div>
```

以上代码包含一个 div，其中含有一个标题和一个段落，我们希望将四张不同的圆角图片分别指定到 div 的四个角上，但是每个对象的背景只能指定一张背景图，为了达到此目的，我们需添加额外的元素用来放置另外三张背景图。于是我们对以上代码做如下修改：

```
<div class="tl">
    <div class="tr">
    <div class="bl">
    <div class="br">
        <h3>&bull; 圆角样式 &bull;</h3>
        <p>
      圆角样式经常用于网站设计当中，使用它可以提高页面的美观性，使页面边界元素看上去更柔和。
        </p>
    </div>
    </div>
    </div>
</div>
```

现在 4 个 div 元素嵌套在一起，里面含有原来的标题和段落，通过 class 属性对每个 div 元素分别设置四个不同的圆角图像，代码如下：

```
div.tl{
    background:url(images/topleft.gif) top left no-repeat;
}
div.tr{
    background:url(images/topright.gif) top right no-repeat;
}
div.bl{
    background:url(images/bottomleft.gif) bottom left no-repeat;
}
div.br{
    background:url(images/bottomright.gif) bottom right no-repeat;
}
```



从浏览器中预览一下效果，发现 4 个圆角已经被添加到指定的位置上(见图 11-43)。

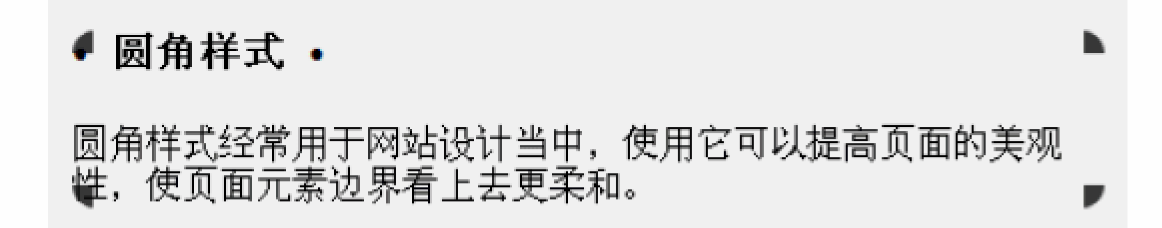


图 11-43　通过 background 属性为 4 个 div 元素添加圆角图片

正确放置好圆角图片之后，我们继续添加其他样式，完善效果：

```
div.tl{
    width:220px;        /* 控制整个元素的宽度 */
    background:#464646 url(images/topleft.gif) top left no-repeat;
}
div.tr{
    background:url(images/topright.gif) top right no-repeat;
}
div.bl{
    background:url(images/bottomleft.gif) bottom left no-repeat;
}
div.br{
    background:url(images/bottomright.gif) bottom right no-repeat;
    padding:10px;       /* 控制其中填充大小 */
}
```

最外侧的 div 元素将背景色设置成为与圆角图像相符的颜色，添加 width 属性以控制整个区域的宽度，而最内侧的 div 控制区域内部与其中内容之间的填充大小。

最后我们再给 h3 和 p 元素添加一些样式，对文本适当地进行排版：

```
h3{
    font-size:1em;
    color:white;
    margin-top:0;
    margin-bottom:8px;
}
p{
    font-size:0.9em;
    line-height:1.8em;
    text-indent:2em;
    color:white;
    padding:0;
    margin:1px 0;
}
```

打开浏览器观察一下效果(见图 11-44)，还不错吧。



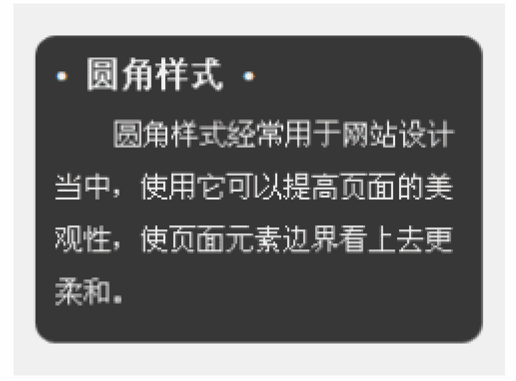


图 11-44　通过嵌套元素实现圆角样式设计

延伸： 在本例中，我们使用了 4 张图像应用到 4 个 div 元素的背景属性中，这样做可以获得最大的灵活性，区域的宽度和高度都能自由变化。对于宽度或高度固定的区域，我们可以将某两个圆角图像进行合并，从而能减少嵌套元素的使用，使页面结构更简洁。

11.4.3　图像替换

Fahrner 图像替换(Fahrner Image Replacement，FIR)是一种 Web 设计方法，由 Todd Fahrner 提出。它使用 CSS 以含有文字的图片替换掉 Web 页面的原始文本。这样做保证了那些不支持 CSS 的浏览器能够正常显示基本的文字信息，同时也不影响搜索引擎对信息的搜索，而对于那些支持 CSS 的浏览器，文字的位置将显示替换的背景图像，从而达到更好的显示效果。

首先我们来看 icebrrg(http://www.icebrrg.com)的 logo 是如何制作的，图 11-45 为 icebrrg 首页的效果图，可以看出该站点使用图像来展示网站标志。



图 11-45　icebrrg 首页



页面中 logo 部分的 XHTML 代码如下：

```
<h1><a href="/" title="Website Title">Icebrrg - Web forms made chillingly
simple</a></h1>
```

h1 元素中含有一个 a 元素，a 元素其中包含了一段文字。下面是相关的 CSS 代码：

```
#header h1{
    background:transparent url(i/icebrrg_logo.jpg) no-repeat scroll left top;
    float:left;
    margin-top:26px;
    text-indent:-9999px;    /* 将文字移除可见区域之外 */
    width:279px;
}
```

以上代码为 h1 元素添加了背景图，同时将 text-indent 设置了一个较大的负数值，使得原来的文本移出了可见区域之外。假如浏览器不支持 CSS，那么浏览器将显示 a 元素内的文本而不是图片(见图 11-46)。



图 11-46　不支持 CSS 的浏览器的显示效果

Superawesome(http://sprawsm.com/)网站的导航也用相同的处理方式(见图 11-47 和 11-48)。

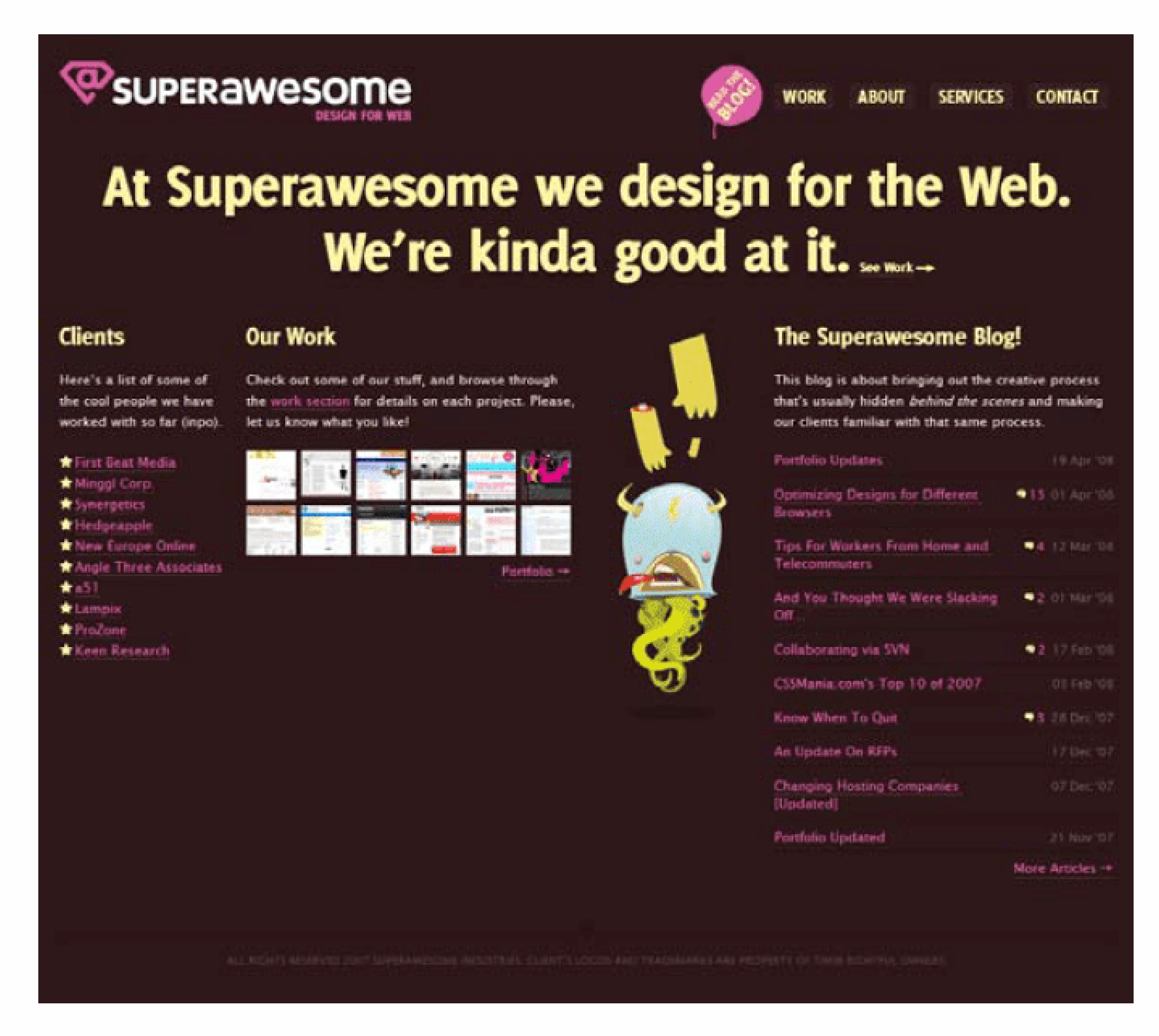
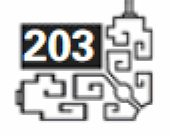


图 11-47　Superawesome 首页



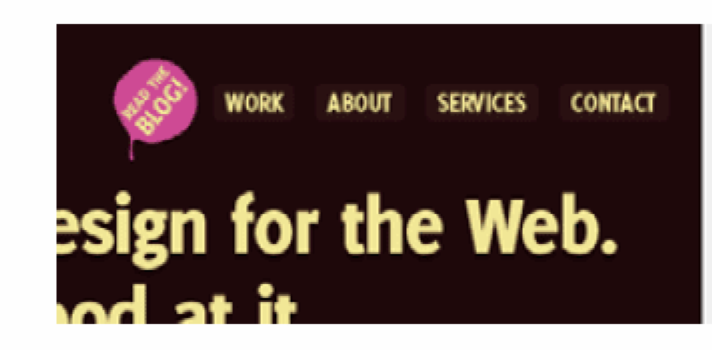


图 11-48　正常情况下的导航菜单(左)和禁用 CSS 后的导航菜单(右)

我们来分析一下该页面的导航菜单是如何实现的，其中的 XHTML 代码如下：

```
<ul id="nav">
   <li class="navWork">
   <a href="http://sprawsm.com/work/" title="See our portfolio">Work</a>
   </li>
   <li class="navAbout">
   <a href="http://sprawsm.com/about/" title="Know more about us">About</a>
   </li>
   <li class="navServices">
   <a href="http://sprawsm.com/services/" title="Design services that we are
offering">Services
   </a>
   </li>
   <li class="blog">
   <a href="http://sprawsm.com/blog/" title="Read the company blog">
   Company blog
   </a>
   </li>
   <li class="navContact">
   <a href="http://sprawsm.com/contact/" title="Holler!">Contact</a>
   </li>
</ul>
```

导航菜单由 ul、li 和 a 元素组成，我们在第 12 章将会详细讲解列表元素的含义和使用方法，这里我们只关注设计者是如何运用 CSS 将文字替换成图片的。添加到第一个 li 中的 a 元素的样式代码如下：

```
/* 第一项菜单的样式 */
#nav li.navWork a{
    background:transparent url(../images/nav work.png) no-repeat scroll 0pt;
    width:54px;
}

/* 链接通用样式 */
#nav li a:link, #nav li a:visited, #nav li a:hover{
    border:medium none;
    display:block;
    height:24px;
    line-height:24px;
    text-indent:-9999px;    /* 将文字移除可见区域之外 */
}
```



链接的通用样式将 a 元素内的文字移到可视区域之外，然后为每个菜单项中的 a 元素添加不同的背景图片。

以上两个站点都使用 text-indent 属性将文字隐藏，类似地，你也可以使用 display:none 或者 visibility:hidden 达到相同的目的。

这种图像替换技术也不是十全十美的，假如浏览器支持 CSS 但关闭了图像显示，那么相应的区域就会是空白。Tom Gilder 和 Levin Alexander 提出的替换方法可以解决这个问题，它的原理是将图像盖在文本上而不隐藏文本。这样当图像被禁用时，下面的文本就会显示在页面上。这种技术也存在局限性，那就是图像不能含有透明信息，否则会露出下面的文本。读者可以访问 http://levinalex.net/files/20030809/alternatefir.html 获得详细信息。

11.5 小　　结

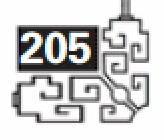
JPEG、GIF 和 PNG 是网络中最常见的三种图像格式，它们有各自的优缺点和适用范围。

使用 img 元素可以向 Web 文档中添加图像，使其作为文档内容的一部分。利用盒模型相关的样式属性可以给页面的图像添加修饰，使用 float 属性可以实现图文混排。

利用 CSS 的背景相关属性可以向页面添加背景和背景图像，这些属性可通过 background 属性进行简化。

最后本章的实例部分介绍了网络相册的制作、圆角技术的应用和 Fahrner 图像替换技术。

在下一章中，将介绍 CSS 在列表元素中的应用。



第 12 章　列　　表

列表在网页中的用途非常广泛，它可以实现文章/新闻的标题列表、图像列表以及导航菜单等。本章将介绍(X)HTML 中与列表相关的各种元素，以及如何利用相关 CSS 属性控制它们的样式。最后将介绍如何利用列表设计新闻标题列表和导航菜单，其中导航菜单涉及了 CSS 设计中的滑动门技术。

本章主要内容

- 列表的种类及用途
- 如何格式化显示列表
- 使用 CSS 更改列表符号的外观
- 如何用列表结合链接实现导航菜单
- 滑动门技术

12.1　列 表 元 素

除了文本段落，许多 Web 内容是以列表形式来呈现信息的。比如新闻标题、导航菜单、术语定义等。通过列表来传达这些信息是最有效、最直接的方法。读者可以浏览一些基于 Web 标准构建的站点，查看其源代码，可以发现导航菜单、部分按钮都是使用列表元素构建的。如图 12-1 所示为 amazon.com 和中国雅虎的页面中使用列表的部分。



图 12-1　列表实例：左图为 amazon.com 首页的导航菜单，右图为中国雅虎首页的一部分

(X)HTML 提供了 3 种列表类型，它们是无序列表(Unordered List)、有序列表(Ordered List)和定义列表(Definition List)，对应的(X)HTML 元素分别为 ul、ol 和 dl。无序和有序列表中每一个列表项用 li 元素表示，定义列表则使用 dt 元素表示术语(Term)项，使用 dd 元素表示定义(Definition)项。

1. 无序列表

无序列表中各表项之间没有特定的顺序关系，比如一份购物清单、某产品的功能介绍等。

Web 页面中常见的“相关链接”就可用无序列表实现。默认情况下，浏览器会为每个列表项之前添加一个项目标记(Marker)。

使用 ul 元素可创建无序列表，每个列表项可使用 li 元素进行创建。

2. 有序列表

有序列表中各表项之间有顺序关系，比如最近的新闻列表或者一系列的操作步骤。有序列表的显示方式与无序列表区别在于，有序列表中的表项之前使用的是序号而不是普通的项目标记。

使用 ol 元素可创建有序列表，每个列表项可使用 li 元素进行创建。

3. 定义列表

定义列表是由成对出现的内容组成：术语项和定义项。类似于字典或百科全书中词条和对应的解释内容。

使用 dl 元素可创建定义列表，使用 dt 和 dd 分别创建术语项和定义项。

下面的代码分别展示了这三类不同的列表：

```
<h3>无序列表：知名汽车品牌</h3>
<ul>
    <li>奔驰</li>
    <li>通用</li>
    <li>丰田</li>
</ul>
<h3>有序列表：NBA 历史得分榜</h3>
<ol>
    <li>贾巴尔 38387 分</li>
    <li>卡尔-马龙 36928 分</li>
    <li>迈克尔-乔丹 32292 分</li>
</ol>
<h3>定义列表：Web 相关术语</h3>
<dl>
    <dt>CSS</dt>
    <dd>Cascading Style Sheet：层叠样式表</dd>
    <dt>HTML</dt>
    <dd>Hypertext Markup Language：超文本标记语言</dd>
    <dt>DOM</dt>
    <dd>Document Object Model：文档对象模型</dd>
</dl>
```

如图 12-2 所示为以上代码的效果。

尽管我们尚未添加任何样式，浏览器还是以不同的方式来显示这三类列表。其中无序列表中每个项目前有实心圆点作为项目标记；有序列表被编上了序号；定义列表中的术语项和定义项采用了不同的缩进以示区别。

无序列表：知名汽车品牌

- 奔驰
- 通用
- 丰田

有序列表：NBA历史得分榜

1. 贾巴尔 38387分
2. 卡尔-马龙 36928分
3. 迈克尔-乔丹 32292分

定义列表：Web相关术语

CSS
    Cascading Style Sheet：层叠样式表
HTML
    Hypertext Markup Language：超文本标记语言
DOM
    Document Object Model：文档对象模型

图 12-2　不同类型的列表

现在再来看一些嵌套使用列表，代码如下：

```
<h3>无序列表：知名汽车品牌</h3>
<ul>
    <li>通用
    <ul>
        <li>凯迪拉克
            <ul>
                <li>CTS</li>
                <li>SRX</li>
                <li>XLR</li>
            </ul>
        </li>
        <li>别克</li>
        <li>雪弗莱</li>
        <li>萨博</li>
    </ul>
    </li>
</ul>
<h3>有序列表：NBA 历史得分榜</h3>
<ol>
    <li>贾巴尔 38387 分
        <ol>
            <li>1962 年 3 月 2 日的 100 分</li>
            <li>1961 年 12 月 8 日的 78 分</li>
            <li>1962 年 1 月 13 日的 73 分</li>
        </ol>
    </li>
</ol>
```

效果如图 12-3 所示。



无序列表：知名汽车品牌

- 通用
  - 凯迪拉克
    - CTS
    - SRX
    - XLR
  - 别克
  - 雪弗莱
  - 萨博

有序列表：NBA历史得分榜

1. 贾巴尔 38387分
   1. 1962年3月2日的100分
   2. 1961年12月8日的78分
   3. 1962年1月13日的73分

图 12-3　嵌套列表

我们看到在嵌套使用无序列表的时候，浏览器会给不同嵌套层次的列表项增加不同的标记，第一层为实心圆，第二层为空心圆，第三层为实心方形。嵌套的有序列表的列表标记仍然是十进制数字。

12.2　列表相关样式

12.2.1　列表样式类型

这里说的类型不是指列表类型，而是每个列表项之前的项目标记的类型。图 12-2 中无序列表表项之前就有实心圆圈作为项目标记，而有序列表中的标记是项目的序号。CSS 提供的 list-style-type 属性可用来修改项目标记的样式。通常用于 ul 和 ol 元素，对于 ul 元素，该属性默认值为 disc；对于 ol 元素默认值则为 decimal。

表 12-1 总结了 list-style-type 的各种属性值和它们对应的含义。

表 12-1　list-style-type 各属性值的含义

属 性 值	含　义
circle	空心圆圈标记(○)
decimal	十进制数字标记(1、2、3、...)
decimal-leading-zero	十进制数字标记，开头加 0(01、02、03、...)
disc	实心圆圈标记(●)
lower-alpha	小写字母标记(a、b、c、...)
lower-roman	小写罗马数字标记(i、ii、iii、...)
none	不显示标记
square	正方形标记(■)
upper-alpha	大写字母标记(A、B、C、...)
upper-roman	大写罗马数字标记(I、II、III、...)

我们对上例的嵌套列表添加如下样式：

```
ul{
    list-style-type:square;
}
ul ul{
    list-style-type:disc;
}
ul ul ul{
    list-style-type:circle;
}
ol{
    list-style-type:upper-roman;
}
ol ol{
    list-style-type:lower-roman;
}
```

效果如图 12-4 所示，注意到无序列表的第一级列表项使用方块、第二级使用实心圆圈、第三级使用空心圆圈作为项目标记。

有序列表采用罗马数字标记时，数字采用右对齐方式。

无序列表：知名汽车品牌

- 通用
  - 凯迪拉克
    - CTS
    - SRX
    - XLR
  - 别克
  - 雪弗莱
  - 萨博

有序列表：NBA历史得分榜

I. 贾巴尔 38387分
   i. 1962年3月2日的100分
   ii. 1961年12月8日的78分
   iii. 1962年1月13日的73分

图 12-4 使用 list-style-type 属性更改默认的项目标记

提示： 对于 circle、disc 和 square，不同浏览器的显示会存在细微的差别。另外，标记和列表项内容使用相同的样式属性，比如字体、颜色等。假如想分别设置二者的样式，可以将列表项单独放在一个内联元素中，比如 span。

除了使用通用的阿拉伯数字、罗马数字和英文字符设置有序列表标记外，list-style-type 还允许设置与具体语言相关的序数符号。具体属性值和含义如表 12-2 所示。

表 12-2　list-style-type 属性值的含义

属 性 值	含　义
armenian	亚美尼亚数字
cjk-ideographic	表意数字(中文、日文、韩文等亚洲语言)
georgian	格鲁吉亚数字
hebrew	希伯来数字
hiragana	日语平假名数字
hiragana-iroha	日语平假名序号
katakana	日语片假名数字
katakana-iroha	日语片假名序号
lower-greek	小写希腊字母

比如我们将上例有序列表的样式做如下修改：

```
ol ol{
    list-style-type:cjk-ideographic;
}
```

列表会使用汉字进行标记(见图 12-5，Firefox 浏览器)。

有序列表：NBA历史得分榜

I. 贾巴尔 38387分
　一. 1962年3月2日的100分
　二. 1961年12月8日的78分
　三. 1962年1月13日的73分

图 12-5　使用亚洲语言进行标记(Firefox 浏览器)

注意： IE 浏览器只支持 CSS1 中的关键字：disc、circle、square、decimal、lower-roman、upper-roman、lower-alpha、upper-alpha 和 none。

12.2.2　列表样式图像

通过 list-style-type 属性可以给列表添加丰富的标记，但是假如仍然不满意 CSS 提供的这些标记符号，可以使用 list-style-image 属性来添加自定义的图像标记。图 12-6 是作者准备的一张图片(宽为 14px，高为 15px)。

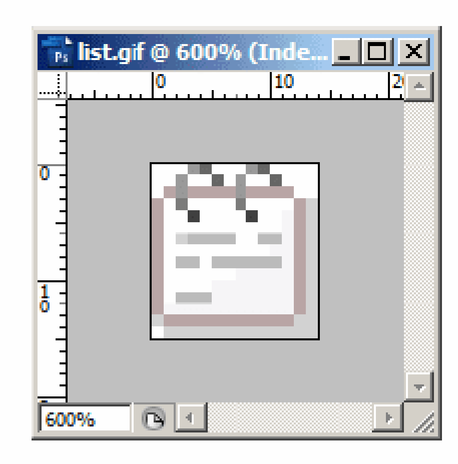


图 12-6　准备列表标记使用的图像

通常列表标记图像与列表中的文字大小相当，否则部分图像会被文字覆盖，影响页面的美观性。接着为 ul 元素添加 list-style-image 属性，代码如下：

```
ul{
    list-style-image:url(images/list.gif);
}

<h3>CSS 中列表样式属性</h3>
<ul>
    <li>list-style-type</li>
    <li>list-style-image</li>
    <li>list-style-position</li>
    <li>list-style</li>
</ul>
```

效果如图 12-7 所示。

CSS中列表样式属性

- list-style-type
- list-style-image
- list-style-position
- list-style

图 12-7　使用 list-style-image 属性添加自定义的项目标记

12.2.3　列表样式位置

虽然使用 list-style-image 属性可以为项目标记指定自定义的图片，但遗憾的是我们不能精确控制图片的位置。CSS 提供的 list-style-position 属性仅仅让我们指定标记出现在列表内容区域之内还是之外，对应的属性值分别为 inside 和 outside。

现将上例的样式和(X)HTML 做如下修改：

```
ul{
    list-style-image:url(images/list.gif);
    border:1px solid black;
    list-style-position:inside;
}
li.outside{
    list-style-position:outside;
}

<h3>CSS 中列表样式属性</h3>
<ul>
    <li>list-style-type</li>
    <li>list-style-image</li>
    <li class="outside">list-style-position</li>
    <li>list-style</li>
</ul>
```

效果如图 12-8 所示。



图 12-8　使用 list-style-position 属性设定标记位置

给 ul 添加边框为的是更容易看出列表区域的范围，从图 12-8 中可以看出第三项标记位于列表区域的外侧，其余三项标记均位于列表区域的内侧。

提示：如果想使用 CSS 精确控制位于项目之前的图片，最好使用 list-style-type:none 来取消默认的项目标记，再用 background 属性为每个 li 元素添加背景图片，最后设置适当的 padding 值让背景图像显示在列表文字的左侧。我们在本章实战部分将采用这种更为灵活的方式来添加自定义的项目标记。

12.2.4　list-style 属性

CSS 提供了 list-style 属性，它可以同时设置多个列表相关的 CSS 属性。比如下列两条 CSS 规则的效果是一致的：

```
ul li{
    list-style-type:none;
    list-style-image:url("list.gif");
    list-style-position:inside;
}

ul li{
    list-style:none url("list.gif") inside;
}
```

12.3　其他相关样式

上面介绍的是列表专用的样式属性，在本节我们介绍一些较为常用的其他与列表相关的样式。

12.3.1　边框、填充和边距

盒模型的这三个属性都可以用在列表元素中，默认情况下，列表是包含一定的填充和边距的。比如下列代码会产生如图 12-9 所示的效果(左边为 IE 浏览器，右边为 Firefox 浏览器)：



```
body{
    margin:0;
    padding:0;
}
ul{
    border:1px solid gray;
}

<ul>
   <li>项目 1</li>
   <li>项目 2</li>
   <li>项目 3</li>
</ul>
```

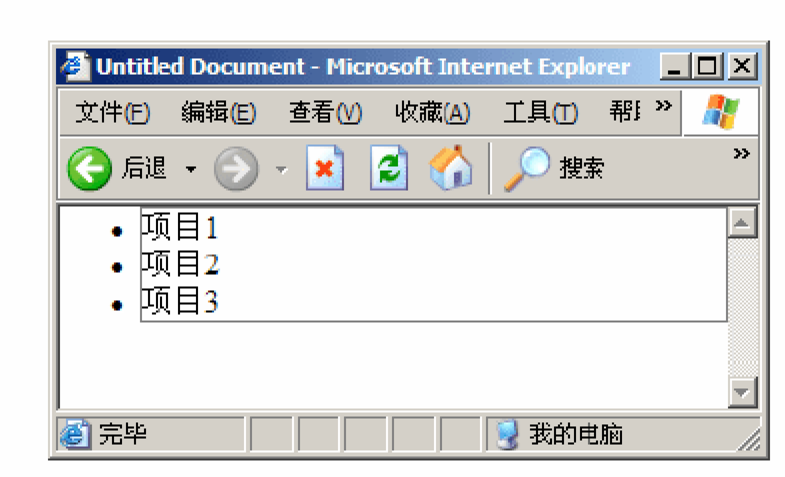
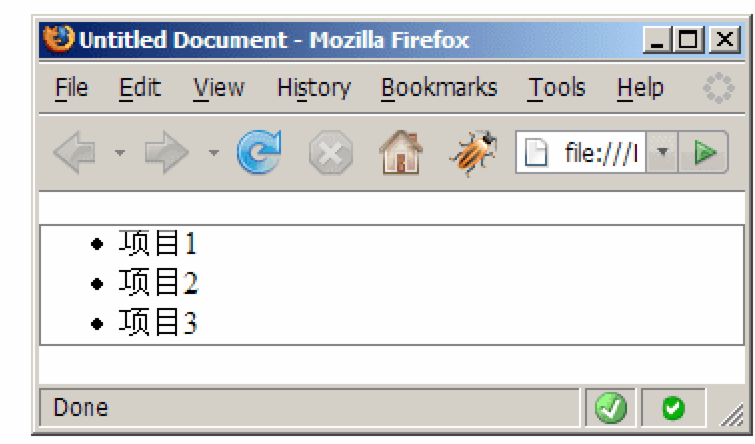



图 12-9　添加了边框的列表(左侧为 IE 浏览器，右侧为 Firefox 浏览器)

在以上两个浏览器中，列表项距离浏览器左边界都有一定的距离，但它们分属于不同的属性控制。从边框的位置可以看出，在水平方向上，IE 浏览器将标记放置在 border 外侧，所产生的距离是由 margin 属性控制的，而 Firefox 中标记位于 border 内，所产生的距离是由 padding 属性产生的。在竖直方向上，IE 中的列表上边框与浏览器之间没有空隙，Firefox 中的列表上边框与浏览器存在一定的距离，这个距离显然是由 margin-top 属性控制的。为了能让间距在两个浏览器中完全去掉，必须将 margin 和 padding 属性都设为 0。CSS 代码如下：

```
ul{
    border:1px solid gray;
    padding:0;
    margin:0;
}
```

图 12-10 所示为最终效果。

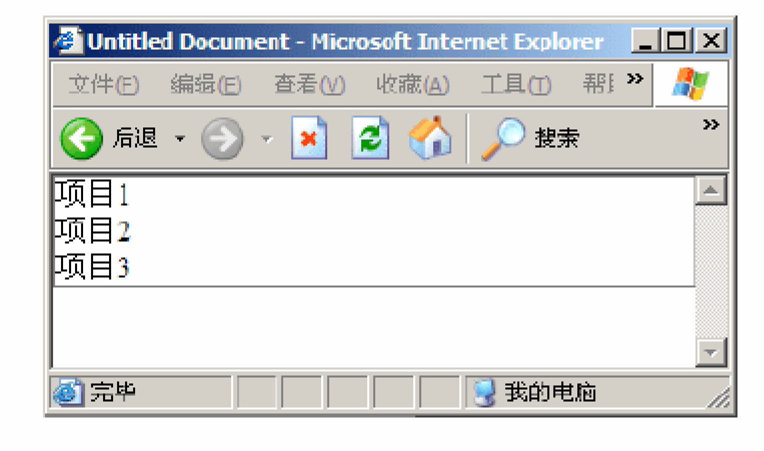
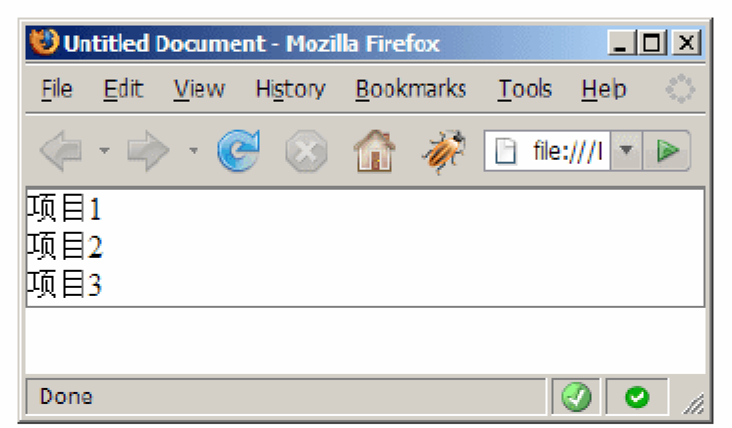



图 12-10　去掉 margin 和 padding 属性(左图为 IE 浏览器，右图为 Firefox 浏览器)



12.3.2　更改布局方式

由于 li 元素为块级元素，列表通常会以垂直方式排列，每个列表项会独占一行。但有时候我们需要用列表实现水平导航菜单，列表项之间不产生换行。一般可用如下两种方法取消列表项之间的换行：

- 更改 li 元素的 display 属性为 inline。默认情况下，li 元素的 display 属性值为 block，把它更改为 inline，使其成为内联元素。但是内联元素会失去一些块级元素的特性，不能设置它的高度、竖直方向上的填充等。
- 更改每个 li 元素的 float 属性为 left。li 元素向左浮动，换行被消除。li 元素依旧保持块级元素的特性。但是浮动控制起来比较复杂，有时元素嵌套过多会产生新的问题。

下面的代码将 li 元素的 display 属性设为 inline，使得列表项之间不存在换行：

```
ul{
    margin:0;
    padding:0;
    list-style:none;
}
ul li{
    display:inline;
}

<ul>
    <li>首页</li>
    <li>新闻</li>
    <li>产品</li>
    <li>论坛</li>
</ul>
```

效果如图 12-11 所示。

首页 新闻 产品 论坛

图 12-11　将 li 元素的 display 属性设为 inline 可取消列表项之间的换行

注意： 不论使用 display:inline 还是 float:left 取消列表项之间的换行，列表标记都将不显示。

12.4　列 表 实 战

12.4.1　新闻列表

新闻列表在 Web 页面中出现的频率相当高，新闻列表可用无序列表或者有序列表来实现，本实战采用无序列表实现一个新闻列表。



首先搭建好新闻列表所需的(X)HTML 代码：

```
<div id="news" class="list">
    <h3>近期资讯</h3>
    <ul>
        <li><a href="1.htm">微软将在 2008 年上半年发布 IE 8 的测试版</a></li>
        <li><a href="2.htm">图书《超越 CSS：WEB 设计艺术精髓》上市</a></li>
        <li><a href="3.htm">微软与 Mozilla 浏览器大战将愈演愈烈</a></li>
        <li><a href="4.htm">RIA 大战：Silverlight vs Flex</a></li>
        <li><a href="5.htm">Mozilla Firefox 3.0 Beta 2 简体中文版发布</a></li>
    </ul>
</div>
```

如图 12-12 所示为不加任何样式的效果。

近期资讯

- 微软将在2008年上半年发布IE 8 的测试版
- 图书《超越CSS：WEB设计艺术精髓》上市
- 微软与Mozilla浏览器大战将愈演愈烈
- RIA大战：Silverlight vs Flex
- Mozilla Firefox 3.0 Beta 2 简体中文版发布

图 12-12　列表的默认效果

首先给 body、div 元素设置如下样式：

```
body{
    margin:10px;
    font-family:"Times New Roman", "宋体", serif;
}
div.list{
    font-size:12px;
}
div.list h3{
    font-size:1.2em;
}
div#news{
    width:280px;
}
```

我们让 body 元素保留 10px 的边距，设置 div 及其 h3 元素中文字的大小，最后设定整个列表宽度为 280px。效果如图 12-13 所示。

近期资讯

- 微软将在2008年上半年发布IE 8 的测试版
- 图书《超越CSS：WEB设计艺术精髓》上市
- 微软与Mozilla浏览器大战将愈演愈烈
- RIA大战：Silverlight vs Flex
- Mozilla Firefox 3.0 Beta 2 简体中文版发布

图 12-13　设置 body、div、h3 元素样式

下面给列表元素和其中的链接元素添加如下的样式：

```
div.list ul{
    list-style:none;
    margin:0;
    padding:0;
    background:#EFEFEF;
}
div.list ul li{
    line-height:1.8em;
}
div.list ul li a{
    color:#000066;
    text-decoration:none;
}
div.list ul li a:hover{
    color:#0066CC;
    text-decoration:underline;
}
```

以上代码取消了浏览器对列表添加的默认样式，除去所有的边距和填充，并设置了背景色。将 li 元素的行距设为 1.8em。最后指定链接的颜色和下划线。效果如图 12-14 所示。

近期资讯

微软将在2008年上半年发布IE 8的测试版
图书《超越CSS：WEB设计艺术精髓》上市
微软与Mozilla浏览器大战将愈演愈烈
RIA大战：Silverlight vs Flex
Mozilla Firefox 3.0 Beta 2 简体中文版发布

图 12-14　除去默认样式、设置行间距后的效果

为了进一步美化外观，要求每条新闻标题之前有图标装饰，行与行之间有线条分隔。图片如图 12-15 所示。

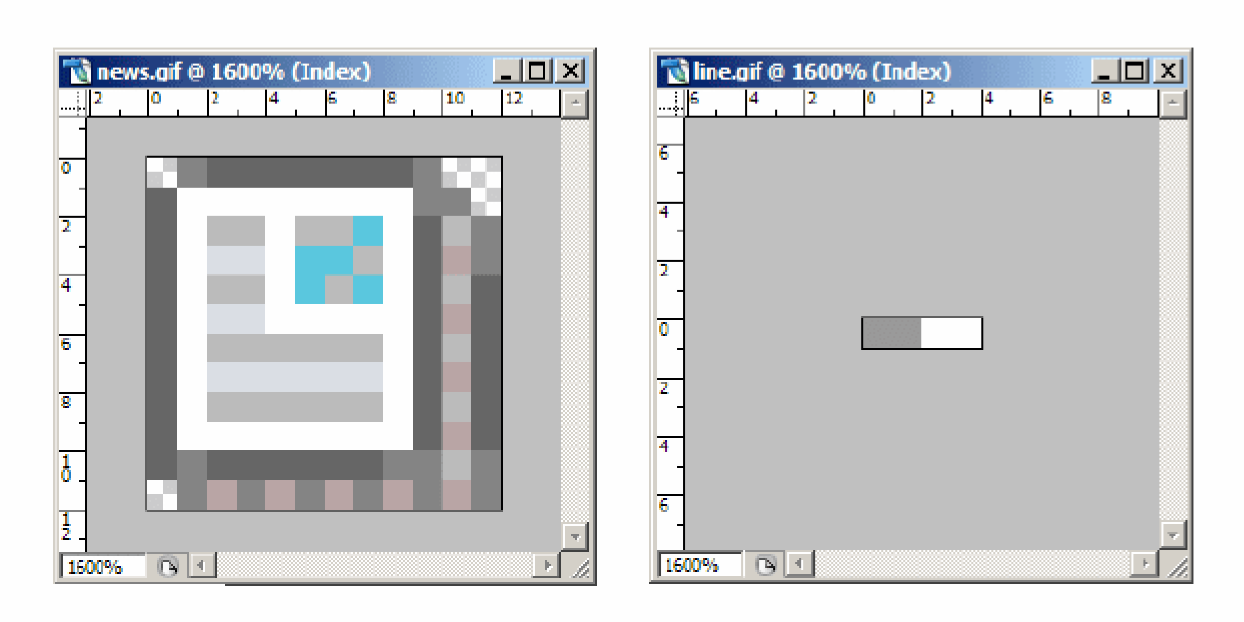


图 12-15　图标和分隔线图片

标题之前的图标可以使用 a 元素中的背景属性添加，而各行之间的线条可通过 li 元素的背景属性添加。于是我们在 a 元素中添加如下声明：

```
background:url(images/news.gif) no-repeat left 1px;
padding-left:16px;
```

而在 li 元素中添加的声明如下：

```
background:url(images/line.gif) repeat-x left bottom;
```

效果如图 12-16 所示。



图 12-16　添加自定义标记和分隔线

我们并不想让分隔线贯穿整个列表区域，想让左右各空出一部分。给 ul 元素添加一些填充即可：

```
padding:0 10px 4px 10px;
```

最后再调整一下标题的样式属性，完整的样式代码如下：

```
body{
    margin:10px;
    font-family:"Times New Roman", "宋体", serif;
}
div.list{
    font-size:12px;
}
div.list h3{
    font-size:1.2em;
    margin:6px 0;
    padding-left:8px;
}
div#news{
    width:280px;
}
/* style for the list */
div.list ul{
    list-style:none;
    margin:0;
    padding:0;
    background:#EFEFEF;
    padding:0 10px 4px 10px;
}
div.list ul li{
    line-height:1.8em;
```



```
    background:url(images/line.gif) repeat-x left bottom;
}
div.list ul li a{
    color:#000066;
    background:url(images/news.gif) no-repeat left 1px;
    text-decoration:none;
    padding-left:16px;
}
div.list ul li a:hover{
    color:#0066CC;
    text-decoration:underline;
}
```

最终效果如图 12-17 所示。



图 12-17　新闻列表最终效果

12.4.2　导航菜单

几乎没有不存在导航菜单的网站，导航菜单通常也是用列表来实现的。图 12-18 展示了几种常见的导航菜单。通过查看源代码会发现，这些导航无一例外地使用了 ul 和 li 元素。



图 12-18　几种常见的导航菜单

延伸： 一个优秀的 Web 导航设计能让网站结构更加清晰、帮助访问者尽快找到所需信息。导航设计涵盖的知识很多，有兴趣的读者可参考 O'Reilly 出版的 Designing Web Navigation 一书。



现在有一份设计好的导航菜单效果图(如图 12-19 所示，较浅的背景表示鼠标悬停在菜单上和当前菜单)，且要求文字大小可依据浏览器的设置进行变化，但不能影响整体效果。我们的任务就是用符合标准的(X)HTML 和 CSS 代码组织成 Web 页面上可用的导航菜单形式。

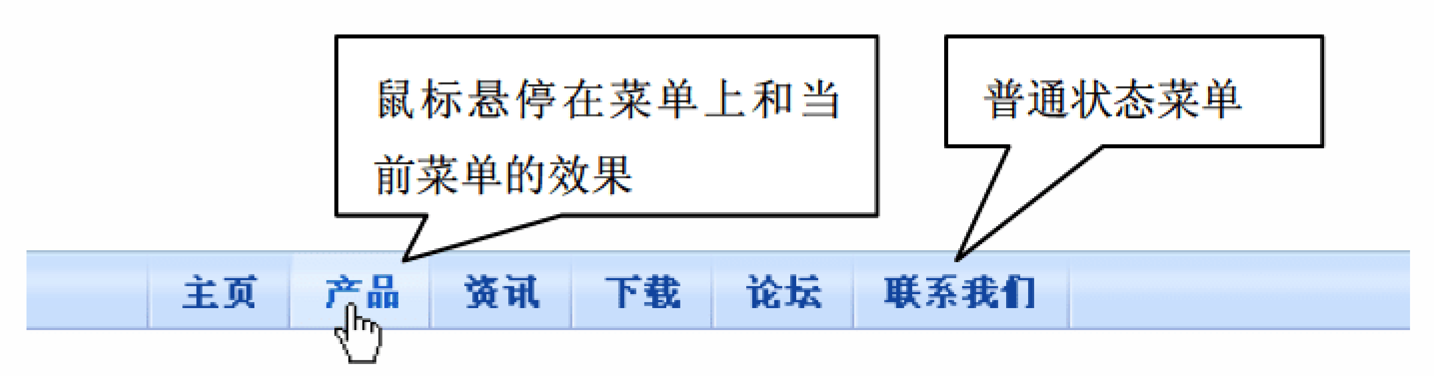


图 12-19　导航菜单效果图

根据效果图和要求，我们首先将图片在竖直方向上进行扩展(见图 12-20)。

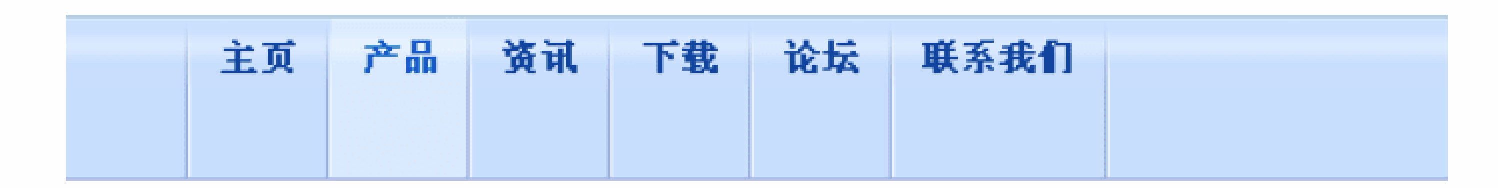


图 12-20　处理过后的导航菜单

从中分割出如下三幅图片(见图 12-21)分别作为导航菜单的背景(nav_bg.gif)、分隔线(nav_divider.gif)和处于悬停、当前菜单状态的背景(nav_highlight.gif)。

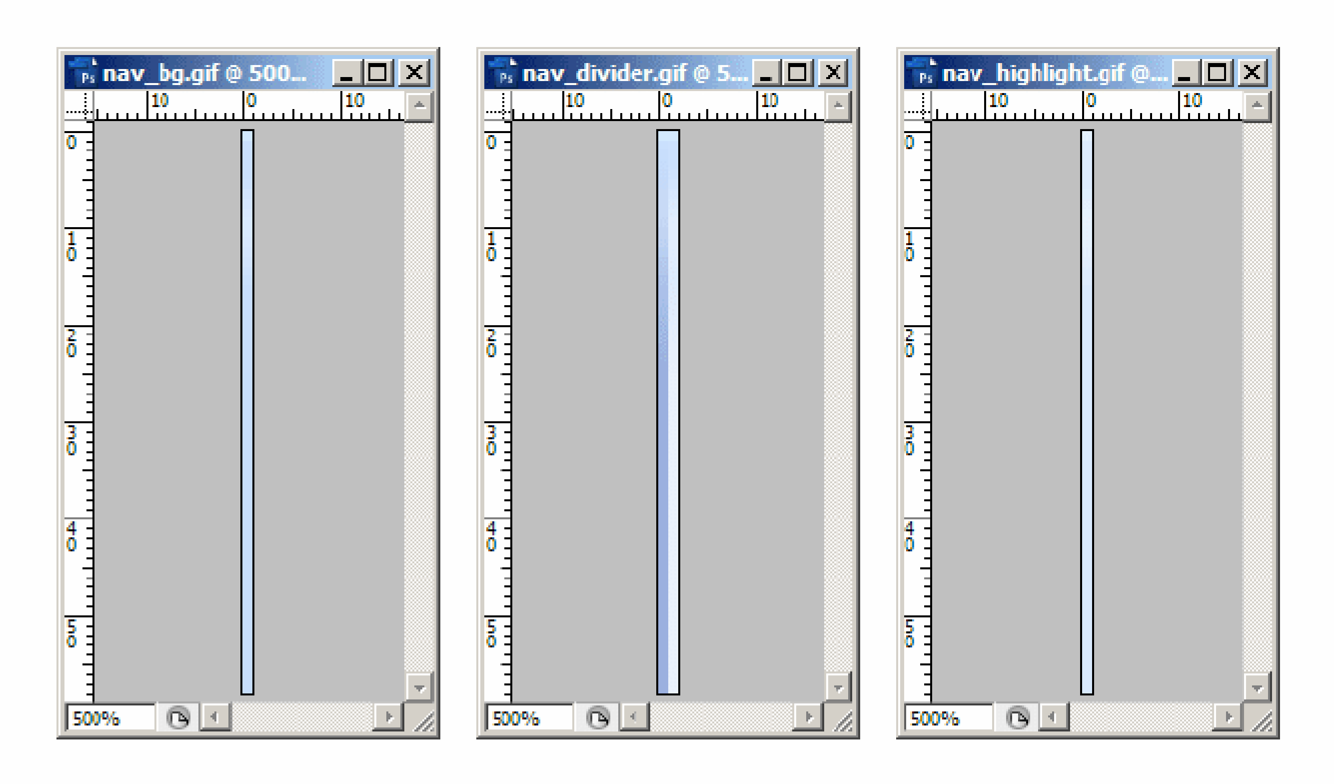


图 12-21　导航菜单的背景和分隔线

下面我们使用列表元素构建导航所需的(X)HTML 代码：

```
<div id="navContainer">
    <ul>
        <li><a href="1.htm">主页</a></li>
        <li><a href="2.htm">产品</a></li>
        <li><a href="3.htm">资讯</a></li>
        <li><a href="4.htm">下载</a></li>
        <li><a href="5.htm">论坛</a></li>
        <li><a href="6.htm">联系我们</a></li>
    </ul>
</div>
```



导航背景放置在 div 元素中，分隔线放置在 li 元素中，表示悬停和当前状态的背景放置在 a 元素中。CSS 代码如下：

```
div#navContainer{
    background:url(images/nav bg.gif) repeat-x left top;
    border-top:1px solid #A7C0D6;
    border-bottom:1px solid #9BAFDE;
    float:left;
}
ul{
    list-style:none;
    padding:0;
    margin:0;
    margin:0 2em;
}
ul li{
    float:left;
    background:url(images/nav_divider.gif) no-repeat left top;
}
ul li a{
    font:bold 0.8em "Times New Roman", "宋体", serif;
    display:block;
    color:#11449E;
    text-decoration:none;
    padding:7px 11px 4px 14px;
}
ul li a:hover{
    background:url(images/nav_highlight.gif) repeat-x -2px top;
}
```

效果如图 12-22 所示。

图 12-22　添加样式后的菜单效果

仔细观察会发现当按钮处于悬停状态时，a 元素的浅色背景遮盖住了左侧的分隔线，这个问题很好解决，我们给每个 li 元素添加 2 个像素的左填充，使里面的 a 元素向右侧移动，让分隔线显露出来。同时，为了让文字与左分隔线的距离不变，a 元素的左填充需要减少 2 个像素，这样就达到平衡了。需要添加和修改的代码如下：

```
ul li{
    padding-left:2px;              /* display the divider */
}
ul li a{
    padding:7px 11px 4px 12px;
}
```

由于每个 li 元素中只有一张分隔线的图片，因此最后一项菜单的右侧还缺少一条分隔线。我们可采用如下办法解决这个问题，在最后一项菜单的右侧再增加一幅背景图片，span 元素的 background 属性已经负责显示其左侧的分隔线，而 a 元素也要负责显示菜单处于 hover 状态时的背景图，因此考虑增加新的(X)HTML 元素：

```
<li><span class="divider"><a href="6.htm">联系我们</a></span></li>
```

然后添加如下样式代码：

```
span.divider{
    display:block;
    padding-right:2px;
    background:url(images/nav_divider.gif) no-repeat right top;
}
```

display 属性设为 block 之后，span 元素才能按其子元素 a 大小而变化。padding-right 设为 2px，保证显示出背景图像。效果如图 12-23 所示。

图 12-23　增加一条分隔线

前面在需求中还提到了要有当前菜单的状态，表示访问者目前浏览的位置。我们添加一个新类 current 表示菜单的“当前”状态，并添加如下样式代码：

```
ul li.current a{
    background:url(images/nav_highlight.gif) repeat-x -2px top;
}
```

只要为 li 元素添加属性 class=“current”就可以使其处于“当前”状态。如图 12-24 所示为将第一个 li 元素的 class 属性设为 current 后的效果。

图 12-24　修改后的菜单效果

注意到我们在设定 a 元素中的字体大小时使用了单位 em，这是个相对大小的单位，表示文字将相对于浏览器默认的文字大小进行变化，现在将浏览器文字大小设为“最大”(依次单击 IE 浏览器菜单中的“查看”→“文字大小”→“最大”命令)，则会出现如图 12-25 所示的效果。

图 12-25　设置文字大小为“最大”后的效果

从图 12-25 中可以看出，文字变大了，导航菜单的高度也随着增大，但由于使用了高度足够大的图片作为元素的背景，才不会出现“撑破”的现象。

延伸： 在本例中，背景图的高度超出了导航菜单实际的高度，这种方法被称作滑动门技术，以这种方式制作的界面能够适应动态缩放或变化的内容。A List Apart 网站上有关于滑动门技术的文章，有兴趣的读者可以访问如下网址：
http://www.alistapart.com/articles/slidingdoors

12.5　小　　结

本章介绍了(X)HTML 中的 3 种列表：无序列表、有序列表和定义列表，列表经常用来实现 Web 页面中的新闻、资源列表、菜单导航等内容，使用频率较高。

列表专用的 3 个 CSS 属性为 list-style-type、list-style-image 和 list-style-position，它们的缩写属性为 list-style。

盒模型相关属性也经常用于控制列表样式，注意 IE 和 Firefox 浏览器中列表默认的 padding 和 margin 属性是不同的。

有两种方法可取消列表项之间的换行，将 display 属性设为 inline，由块级元素改为内联元素；或者使用 float:left 将元素向左浮动。二者各有优缺点。

下一章将介绍表格的使用以及如何用 CSS 美化表格的样式。



第 13 章　表　　格

表格起初被误用作页面布局的工具，但是它的真正用途是用来显示表格化的数据。本章将向读者介绍(X)HTML 中与表格相关的各种元素，以及如何利用它们来组织表格化的数据。接下来将介绍 CSS 中与表格相关的样式属性，如何设置表格的大小、控制表格中文字的对齐方式。在本章的实例部分，将介绍如何制作斑马线效果的表格以及如何使用表格来实现一个在线日历。

本章主要内容

- (X)HTML 中与表格相关的标记
- 正确使用表格
- 表格边框模式和样式
- 控制表格宽度和高度
- 控制表格内容对齐方式

13.1　使 用 表 格

13.1.1　表格用途

在实际生活中，我们经常接触表格，比如通讯录、比赛成绩、课程表等内容都是通过表格的形式呈现出来的。表格如此常见，以至于 Web 页面中不能没有它，(X)HTML 中就包含了创建表格所需要的各种标记。

但是，表格曾经被许多设计者用来规划页面布局，为了达到某种效果，不得不嵌套使用大量的表格，最终导致页面灵活性很差，代码也显得繁琐。

尽管可以用表格来布局，但是这并不是表格正确的用途。表格应该用来展现那些适合表格化显示的信息，而不是作为一种布局工具。

13.1.2　表格元素

与表格相关的元素有 table、colgroup、col、tr、th、td、caption、thead、tbody 和 tfoot，看上去内容不算少，现在就分别进行介绍。

1. 基本表格元素

尽管与表格相关的(X)HTML 元素相当多，但是构建一个最简单的表格只需要 table、tr 和 td 这三个元素。table 元素表示整个表格，tr 表示表格中的一行，td 在 tr 元素内部使用，表示一个单元格。

下面的代码建立了一个两行三列的表格：

```
<table>
    <tr>
        <td>2007-12-12</td>
        <td>活塞 - 灰熊</td>
        <td>113 - 103</td>
    </tr>
    <tr>
        <td>2007-12-12</td>
        <td>猛龙 - 老鹰</td>
        <td>100 - 88</td>
    </tr>
</table>
```

效果如图 13-1 所示。尽管使用的是表格元素，但是大多数浏览器在默认情况下并不显示表格的边框。

2007-12-12	活塞 - 灰熊	113 - 103
2007-12-12	猛龙 - 老鹰	100 - 88

图 13-1　基本表格

基本标记还包括 th 和 caption 元素，th 和 td 类似，也在 tr 内部出现，但 th 表示的是一个表头(Header)而不是普通的单元格。caption 元素包含了表格的标题。我们给上面的代码添加 th 和 caption 元素。注意 caption 元素需要紧跟在<table>标记之后：

```
<table>
    <caption>近期赛况</caption>
    <tr>
        <th>日期</th>
        <th>球队</th>
        <th>比分</th>
    </tr>
    <tr>
        <td>2007-12-12</td>
        <td>活塞 - 灰熊</td>
        <td>113 - 103</td>
    </tr>
    <tr>
        <td>2007-12-12</td>
        <td>猛龙 - 老鹰</td>
        <td>100 - 88</td>
    </tr>
</table>
```

代码效果如图 13-2 所示，表格的标题位于顶端，表头文字显示为粗体并且水平居中于单元格中。



近期赛况		
日期	**球队**	**比分**
2007-12-12	活塞 - 灰熊	113 - 103
2007-12-12	猛龙 - 老鹰	100 - 88

图 13-2　为表格添加 th 和 caption 元素

2. 高级表格标记

(X)HTML 提供了 3 个按行定义表格区域的元素，它们是 thead、tbody 和 tfoot。thead 和 tfoot 分别定义了表头和表底区域，tbody 则可以将表格划分为多个部分，具体如何划分要根据实际需要而定。这种划分是位于逻辑层面上的，目的是使得表格结构更为清晰。

比如我们可以给前面的表格添加三个区域的定义：

```
<table>
    <caption>近期赛况</caption>
    <thead>
        <tr>
            <th>日期</th>
            <th>球队</th>
            <th>比分</th>
        </tr>
    </thead>
    <tfoot>
        <tr>
            <td colspan="3">点击此处查看比赛详情</td>
        </tr>
    </tfoot>
    <tbody>
        <tr>
            <td id="r2c1">2007-12-12</td>
            <td id="r2c2">活塞 - 灰熊</td>
            <td id="r2c3">113 - 103</td>
        </tr>
        <tr>
            <td>2007-12-12</td>
            <td>猛龙 - 老鹰</td>
            <td>100 - 88</td>
        </tr>
    </tbody>
</table>
```

在上面代码中，tfoot 内部的 td 元素使用了 colspan 属性，它的含义是将单元格扩展到多个列，类似地还有 rowspan 属性。可以注意到，尽管 tfoot 元素的代码位于 tbody 之前，但显示的时候总是位于 tbody 之后(见图 13-3)。



近期赛况

日期	球队	比分
2007-12-12	活塞 - 灰熊	113 - 103
2007-12-12	猛龙 - 老鹰	100 - 88
点击此处查看比赛详情		

图 13-3　添加逻辑区域划分

(X)HTML 中的表格是以行为中心的，即在创建表格的过程中先定义行元素，然后在其内部定义每个单元格。当然也可以将表格看作列的集合，使用 colgroup 和 col 元素即可实现。

colgroup 元素可按照列将 table 划分成多个组，假如一个 6 列的表格按列分为两组，可用如下代码：

```
<colgroup id="cg1" span="2"></colgroup>
<colgroup id="cg2" span="4"></colgroup>
```

以上代码将表格前两列和后四列分为两个不同的组，这是由 span 属性确定的。使用 col 元素可将表格分得更细致，每个 col 元素代表表格中的一列。比如：

```
<colgroup>
    <col id="c1" />
    <col id="c2" />
</colgroup>
<colgroup>
    <col id="c3" />
    <col id="c4" />
    <col id="c5" />
    <col id="c6" />
</colgroup>
```

每个表格元素所包含的属性及其具体含义在这里不做详细讲解，需要进一步了解的读者请参考其他相关书籍。

13.2　边 框 样 式

本节将为 13.1 节所示的表格添加样式。与表格关系较为密切的样式为表格的边框、单元格内部的填充、单元格之间的间距、背景色、文字对齐方式等。表格元素中的某些属性(border、cellspacing 等)可完成与 CSS 类似的设定表格样式的功能，但 CSS 控制得更为出色，因此不推荐使用元素的属性，让样式全部由 CSS 管理。

13.2.1　border 属性

边框可以说是表格最常见的样式之一，仿佛没有边框的表格不能算是一个真正的表格。

相信读者已经想到盒模型的 border 属性了，首先给 table 添加 border 属性：



```
table{
    border:1px solid black;
}

<table>
    <caption>近期赛况</caption>
    <tr>
        <th>日期</th>
        <th>球队</th>
        <th>比分</th>
   </tr>
    <tr>
        <td>2007-12-12</td>
        <td>活塞 - 灰熊</td>
        <td>113 - 103</td>
    </tr>
    <tr>
        <td>2007-12-12</td>
        <td>猛龙 - 老鹰</td>
        <td>100 - 88</td>
    </tr>
</table>
```

从图 13-4 看出，给 table 元素添加边框属性后，边框只应用在表格的最外侧，而内部的单元格之间没有边框，另外表格的标题位于边框外侧。

近期赛况

日期	球队	比分
2007-12-12	活塞 - 灰熊	113 - 103
2007-12-12	猛龙 - 老鹰	100 - 88

图 13-4　给 table 元素添加 border 属性

通常表格内部各个单元格之间要用线框分隔开，table 的 border 属性并没有起到作用，于是我们尝试再给表头 th 元素和单元格 td 元素添加 border 属性。

```
th, td{
    border:1px solid black;
}
```

图 13-5 为添加代码之后的效果。

近期赛况

日期	球队	比分
2007-12-12	活塞 - 灰熊	113 - 103
2007-12-12	猛龙 - 老鹰	100 - 88

图 13-5　给 th 和 td 元素添加 border 属性

现在表头和单元格都拥有了自己的边框，假如我们希望表头或单元格之间用 1 个像素宽的单一线框分隔，该如何实现呢？浏览器在默认情况下给单元格添加了间距，使得单元格之间存在空隙。下面的内容将会告诉读者如何用 CSS 控制这个间距。

13.2.2　border-spacing 属性

CSS 的 border-spacing 属性等同于 table 元素的 cellspacing 属性，可用来控制单元格的间距。属性值为一个值的时候表示单元格四个方向上与相邻单元格之间的距离；若属性值包含两个值的时候，则前者表示水平方向上的间距，后者表示竖直方向上的间距。

现在给上面示例中的 table 元素中增加如下声明：

```
border-spacing:1em;
```

从图 13-6(Firefox 浏览器)中可以看出，单元格之间的距离被扩大了。



近期赛况

日期	球队	比分
2007-12-12	活塞 - 灰熊	113 - 103
2007-12-12	猛龙 - 老鹰	100 - 88

图 13-6　以 border-spacing 更改单元格间距(Firefox 浏览器)

若将上面的声明改为：

```
border-spacing:10px 0;
```

则会产生如图 13-7(Firefox 浏览器)所示的效果，单元格水平方向间距为 10px，竖直方向间距为 0。

近期赛况

日期	球队	比分
2007-12-12	活塞 - 灰熊	113 - 103
2007-12-12	猛龙 - 老鹰	100 - 88

图 13-7　border-spacing 可分别控制单元格水平和垂直方向上的间距(Firefox 浏览器)

注意：由于 IE 浏览器不支持 border-spacing 属性，因此只能在 table 元素中添加 cellspacing 属性以达到相同的效果。但 cellspacing 属性没有 border-spacing 灵活，不能分别控制水平和垂直方向的间距。

13.2.3　边框模式

现在可以解决 13.2.1 节中提出的问题了，假如想让上面示例中的单元格之间的线框只有 1 个像素宽，该怎么实现？这涉及到了 CSS 中的边框模式(Border Modal)的概念，针对表格边框



CSS 提供了两种不同的模式：分离(Separated)边框模式和塌陷(Collapsed)边框模式。分离模式意味着表格内每个单元格之间是分开的，当单元格之间距离为 0 时，边框会紧挨在一起，但不会重叠。塌陷模式表示单元格之间是连续的，相邻单元格的边框“塌陷”在一起，即重叠为一条边框。使用 border-collapse 属性可对边框模式进行选择。属性值 separate 和 collapse 分别对应分离模式和塌陷模式。

1. 分离模式

在分离模式下，表格中单元格之间是分离的，即存在一定的空间，如图 13-5 所示的表格就处于该模式下。这是大部分浏览器默认的边框模式。在分离模式下我们可以通过设定 border-spacing 属性来更改单元格间距。

图 13-8 为添加声明 border-spacing:0 后在 Firefox 浏览器中显示的效果，它和设定 cellspacing 属性为 0 的效果相同。

近期赛况

日期	球队	比分
2007-12-12	活塞 - 灰熊	113 - 103
2007-12-12	猛龙 - 老鹰	100 - 88

图 13-8　使用 border-spacing 属性取消单元格之间的距离(Firefox 浏览器)

这样单元格的边框就紧挨在一起了，形成了 2 个像素宽的线框，即形成了所谓的双重边框(Double Borders)。但是如何做到让边框只有 1 个像素宽呢？塌陷模式可以解决这个问题。

2. 塌陷模式

在塌陷模式下，表格中相邻单元格的边框重叠在一起，在实际中要比分离模式的使用广泛。由于相邻单元格共享边框，当 border 属性存在冲突时，要根据以下规则处理：

- 如果 border-style 属性设置为 hidden，其他所有边框样式将不起作用。
- 如果 border-style 属性设置为 none，其他任何边框样式都将起作用。
- 除了将 border-style 属性设为 hidden，较粗的边框将覆盖较细的边框，如果相邻单元格的边框粗细一致，边框样式将按照如下顺序显示：double、solid、dashed、dotted、ridge、outset、groove、inset。
- 如果相邻单元格的边框粗细和样式均一致，但颜色相异，边框颜色将按照如下顺序决定：单元格、行、行组、列、列组、表格。

并不是所有的浏览器都能严格遵循以上规则。在不同浏览器上的实现存在细微的差别，但在一般情况下这种冲突很少发生，因此这里不做过多讨论，感兴趣的读者可以自己进行试验。

现在我们给上一示例中的 table 元素添加如下声明：

```
border-collapse:collapse;
```

效果如图 13-9 所示，这也是我们经常见到的表格样式。



近期赛况		
日期	球队	比分
2007-12-12	活塞 - 灰熊	113 - 103
2007-12-12	猛龙 - 老鹰	100 - 88

图 13-9　使用 border-collapse 属性将相邻边框重叠在一起

13.3　与表格相关的样式

13.3.1　caption-side 属性

表格的标题由 caption 元素确定，默认情况下标题位于表格顶部，使用 caption-side 属性可调整标题的位置。请看如下的示例：

```
table#leftCap{
    border:1px solid red;
    caption-side:left;
}
table#topCap{
    border:1px solid red;
    caption-side:top;
}
table#rightCap{
    border:1px solid red;
    caption-side:right;
}
table#bottomCap{
    border:1px solid red;
    caption-side:bottom;
}

<table id="leftCap">
    <caption>表格标题</caption>
    <tr><td>使用 caption-side 可控制表格标题的位置。</td></tr>
</table>
<table id="topCap">
    <caption>表格标题</caption>
    <tr><td>使用 caption-side 可控制表格标题的位置。</td></tr>
</table>
<table id="rightCap">
    <caption>表格标题</caption>
    <tr><td>使用 caption-side 可控制表格标题的位置。</td></tr>
</table>
<table id="bottomCap">
    <caption>表格标题</caption>
    <tr><td>使用 caption-side 可控制表格标题的位置。</td></tr>
</table>
```



效果如图 13-10 所示(Firefox 浏览器)，需要说明的是，IE 和 Opera 浏览器均不支持该属性，标题总是位于表格顶部。

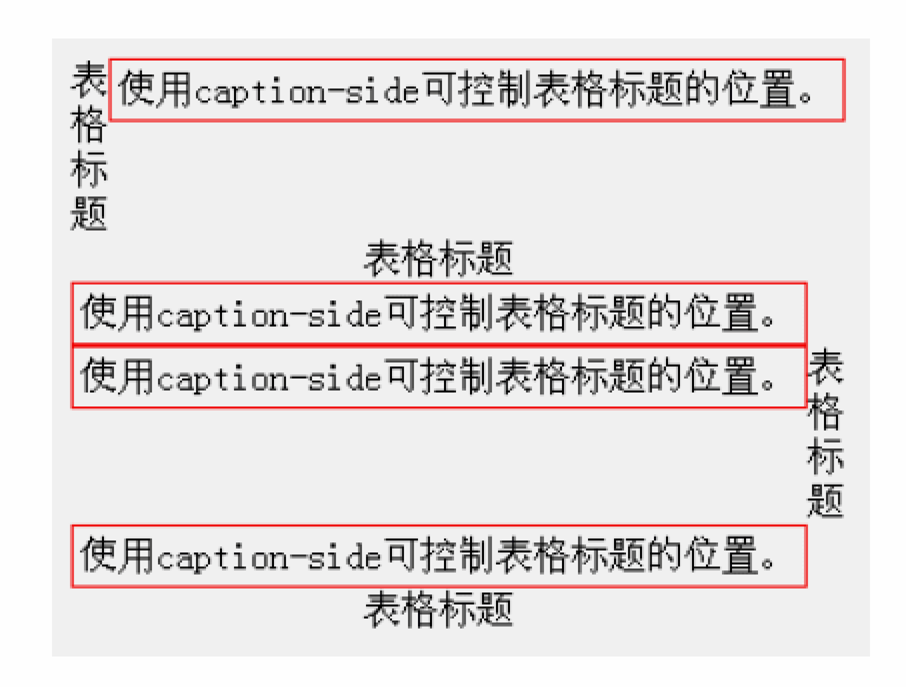


图 13-10 caption-side 设置标题位置(Firefox 浏览器)

13.3.2 添加填充

盒模型中的 padding 属性同样可用在表格上，一般在 th 或 td 元素中使用。padding 在这里表示单元格中的内容与其边框之间的距离。

假如要给赛况表格中的 th 和 td 元素添加一些填充，则只需添加如下代码。

```
padding:0.4em;
```

效果如图 13-11 所示。文字和所在单元格的边框之间的距离变大了。

近期赛况

日期	球队	比分
2007-12-12	活塞 - 灰熊	113 - 103
2007-12-12	猛龙 - 老鹰	100 - 88

图 13-11 添加 padding 属性

除了上述样式外，背景颜色、文字风格、颜色等样式属性均可应用到表格中。

注意：IE 和 Firefox 浏览器对于填充的显示效果不完全相同，同样的 padding 值会在 Firefox 浏览器中产生较多的填充。如果对页面效果精度要求很高的话，在设定 padding 属性时要注意兼顾多个浏览器的效果。

13.4 表格大小

显示在页面中的表格呈矩形，有自己的宽度和高度。由于表格内可能会包含多个单元格，每个单元格又拥有边框、填充等属性，因此宽度控制起来比较复杂。本节将会介绍如何控制表格的宽度及高度。

13.4.1　表格宽度计算方式

表格宽度的计算有两种不同的方式：固定宽度方式和自动宽度方式，而高度只有自动方式。设计者使用 CSS 提供的 table-layout 属性来选择使用哪种方式，属性值 fixed 和 auto 分别表示固定宽度和自动宽度方式。

1. 固定宽度方式

在固定宽度方式中，宽度将根据表格、列和单元格的 width 属性决定，而不是由单元格内容而定。固定宽度方式依照如下规则进行宽度计算。

(1) 任何设置了宽度值的列元素都将按照此值确定宽度。

(2) 如果某一列的宽度设为自动宽度(这是默认情况)，但该列的第一行单元格设置了宽度值，则该列按照此值确定宽度。如果这个单元格跨越多个列，则每个列的宽度取平均值。

(3) 任何设置为自动宽度的列都会自动设定一个宽度值，并保证宽度值是平均分配的。

请看下面的示例：

```
body{
    font-family:"Arial", "宋体", sans-serif;
    font-size:12px;
}
table{
    border:none;
    border-collapse:collapse;
    width:300px;
}
th{
    text-align:left;
    background-color:#FFFF66;
}
th,td{
    border:none;
    padding:0;
}

<table>
   <caption>我的唱片集</caption>
   <colgroup>
       <col id="album" />
       <col id="artist" />
       <col id="released" />
    </colgroup>
    <thead>
        <tr>
```



```
            <th>艺术家</th>
            <th>唱片集</th>
            <th>年份</th>
        </tr>
    </thead>
    <tbody id="chinese">
        <tr>
            <td>孙燕姿</td>
            <td>自选集</td>
            <td>2002</td>
        </tr>
        <tr>
            <td>周杰伦</td>
            <td>叶惠美</td>
            <td>2003</td>
        </tr>
    </tbody>
    <tbody id="western">
        <tr>
            <td>Micheal Jackson</td>
            <td>Invincible</td>
            <td>2001</td>
        </tr>
        <tr>
            <td>Christina Aguilera</td>
            <td>Stripped</td>
            <td>2002</td>
        </tr>
    </tbody>
</table>
```

以上代码产生的效果如图 13-12 所示，表格宽度为 300px。

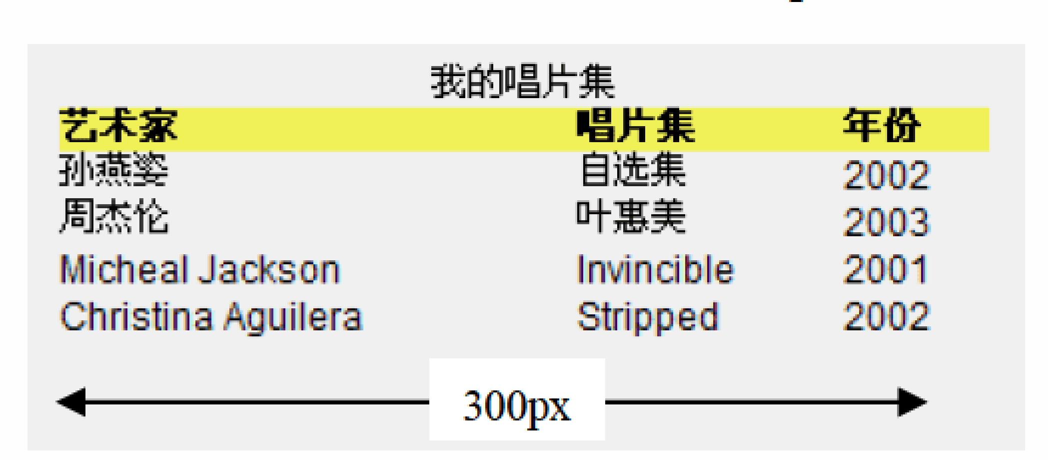


图 13-12 设定宽度为 300 个像素

若将宽度值设为 100px，则产生如图 13-13 所示的效果，表格实际宽度要比 100px 宽，这是由于默认情况下浏览器按自动方式计算表格宽度，即 table-layout 属性值为 auto，这样每个单元格都要完全显示其所包含的内容。



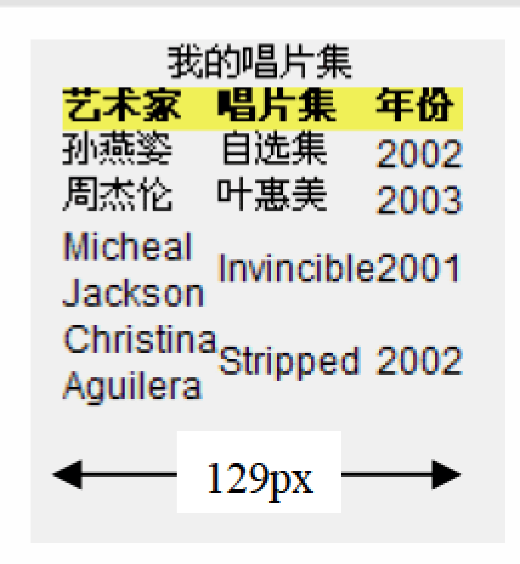


图 13-13　设定宽度为 100 个像素，但实际宽度要大于此值

现在修改表格的宽度计算方式为 fixed，表格会严格按照宽度值进行计算，效果如图 13-14 所示：

```
table-layout:fixed;
```



图 13-14　添加 table-layout:fixed 声明后，表格宽度会严格依照 width 属性值

现在将表格宽度改回至 300px，并修改 table、th 和 td 元素的 border 属性，代码如下：

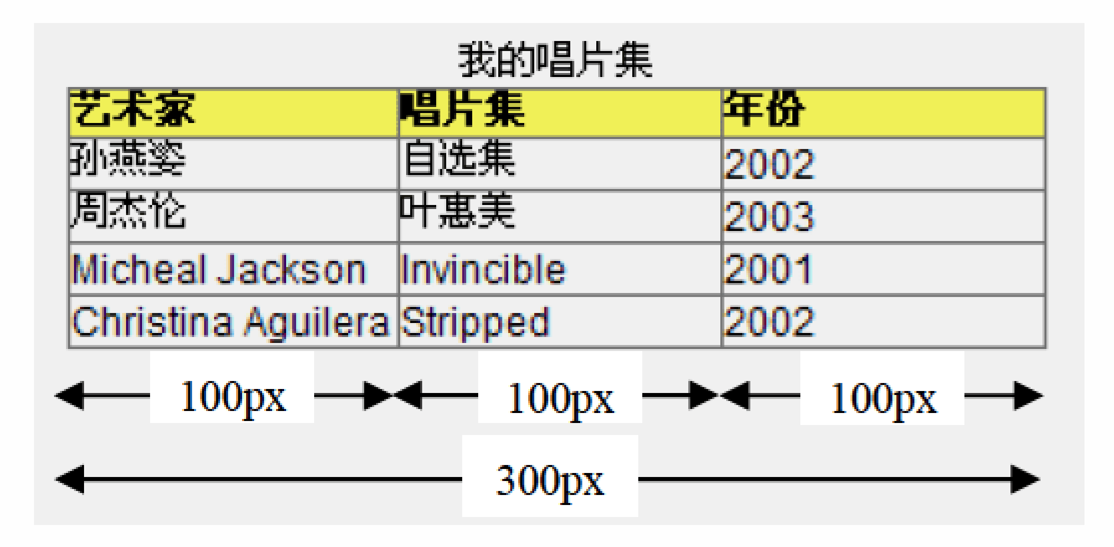
```
border:1px solid gray;
```

效果如图 13-15 所示，它与图 13-12 的区别在于，设定了固定宽度方式后，由于单元格没有设置宽度值，默认为 auto。根据规则 3 的要求，每个单元格的宽度应该相等，即它们平均分配 300px 的宽度，因此每一列的宽度均是 100px。另外，添加边框之后的表格宽度依旧是 300px，边框的宽度不会影响 table 元素的 width 属性。

表格第一列中的内容较多，应该设定一个较宽的宽度值，年份一列最多只有 4 位数字，可指定一个较小的宽度值。于是添加如下规则：

```
col#album{
    width:160px;
}
col#released{
    width:40px;
}
```





我的唱片集

艺术家	唱片集	年份
孙燕姿	自选集	2002
周杰伦	叶惠美	2003
Micheal Jackson	Invincible	2001
Christina Aguilera	Stripped	2002

图 13-15　未设定宽度值的单元格会平均分配表格宽度

以上规则利用 col 元素设定每个列的宽度，余下的一列宽度会自动计算而得，应该是 300px−160px−40px=100px，效果如图 13-16 所示。

我的唱片集

艺术家	唱片集	年份
孙燕姿	自选集	2002
周杰伦	叶惠美	2003
Micheal Jackson	Invincible	2001
Christina Aguilera	Stripped	2002

160px　100px

300px

图 13-16　给 col 元素设定宽度

最后调整一下表格标题和表格之间的距离，并给单元格增加一些填充。需要添加和修改的代码如下：

```
caption{
    padding-bottom:5px;
}

th,td{
    border:1px solid gray;
    padding:4px;
}
```

效果如图 13-17 所示，我们看到，不论是边框还是填充，它们都不会影响表格的最终宽度。

我的唱片集

艺术家	唱片集	年份
孙燕姿	自选集	2002
周杰伦	叶惠美	2003
Micheal Jackson	Invincible	2001
Christina Aguilera	Stripped	2002

300px

图 13-17　添加填充之后，表格宽度不发生改变



2. 自动宽度方式

自动宽度方式是浏览器默认的方式，表格宽度将根据列中单元格内容自动调整宽度值。假如没有指定 table 元素的宽度，则总宽度依据每一列设定的宽度值或者能显示所有内容的最小宽度值计算确定。若指定了 table 的宽度且该宽度大于计算值，则每一列会平均分配赋予出来的宽度，否则 table 的宽度取计算值。

我们将上例中的 table-layout 属性值改为 auto，取消其 width 属性，再添加以下代码设置第二列的宽度为 60 个像素：

```
col#artist{
    width:60px;
}
```

将会产生如图 13-18 所示的效果。

我的唱片集

艺术家	唱片集	年份
孙燕姿	自选集	2002
周杰伦	叶惠美	2003
Micheal Jackson	Invincible	2001
Christina Aguilera	Stripped	2002



图 13-18　自动宽度计算方式

表格最后宽度是 288px，然而三列的宽度值之和是 260px，那么多余出来的 28 个像素是从哪里来的呢？实际上，每一列的宽度是指的其内容区域的宽度，不包括填充和边框部分，因此表格总体宽度应为三列的宽度值之和再加上边框和填充的宽度。这里四条边框宽度值之和为 4px，每个单元格水平方向有 8px 的填充(左右各 4px)，三列一共 24px，所以宽度值应为 260px+4px+24px=288px。这就是自动宽度方式下表格宽度的计算方式。

延伸： 当选择自动宽度方式计算表格宽度时，表格显示的速度要比固定方式慢。这是由于浏览器要等到表格所有内容全部载入之后才能确定整体宽度，而且还要加上边框、填充部分的宽度。假如某一列的宽度值为百分数，则还需要再次计算。

13.4.2　高度

表格的高度计算没有宽度那么复杂，如果设定了 table 的高度值，则按照此值确定高度，每行的高度根据自己的 height 值确定高度，若行高设置为 auto，则根据表格的高度自动选取一个值，且所有高度为 auto 的行的高度一致；如果 table 的高度值为 auto，则总高度根据该列的所有单元格的高度值(设置了具体的 height 值)或内容区域的高度(height 值为 auto)加上边框、填充的大小。另外，给高度值设定百分数不起任何作用。

假如在上例中给 table 元素设置 height 值为 280px，则表格总高度将是 280px，各行的高度

都会适当增加，高度值一致(见图 13-19)。

我的唱片集

艺术家	唱片集	年份
孙燕姿	自选集	2002
周杰伦	叶惠美	2003
Micheal Jackson	Invincible	2001
Christina Aguilera	Stripped	2002

280px

图 13-19　设定 table 元素的高度

现在将 table 的 height 属性设为 auto，给 th 和 td 元素添加 height 属性，并设定其高度为 30px，则表格总高度将是 150px+6px+40px=196px(见图 13-20)。



我的唱片集

艺术家	唱片集	年份
孙燕姿	自选集	2002
周杰伦	叶惠美	2003
Micheal Jackson	Invincible	2001
Christina Aguilera	Stripped	2002

图 13-20　设定 th 和 td 元素的高度

13.5　对　　齐

1. 水平方向对齐

控制水平方向对齐只需要用 text-align 属性即可，上例中我们就用 text-align 属性将 th 元素中的内容设定为左对齐。

2. 垂直方向对齐

要控制单元格中的内容在垂直方向的对齐方式，可以使用 vertical-align 属性，前面讲过，该属性可用来控制文本的垂直对齐方式，但是当它用在表格中时，属性值的含义有所改变。各个属性值的含义如表 13-1 所示。



表 13-1　vertical-align 属性值的含义

属 性 值	含　义
top	单元格中内容与本行顶部对齐
middle	单元格中内容与本行中心位置对齐
bottom	单元格中内容与本行底部对齐
baseline	单元格中内容与本行基线位置对齐
sub、super、text-top 和 text-bottom	这些属性值将被忽略，不起作用

请看如下示例：

```
table{
    width:360px;
    height:200px;
}
td{
    border:1px solid black;
}
td#col1{
    vertical-align:top;
}
td#col2{
    vertical-align:middle;
}
td#col3{
    vertical-align:bottom;
}

<table>
    <tr>
        <td id="col1">本单元格内容将与本行的顶部对齐。</td>
        <td id="col2">本单元格内容将与本行中心位置对齐。</td>
        <td id="col3">本单元格内容将与本行底部对齐。</td>
    </tr>
</table>
```

效果如图 13-21 所示。

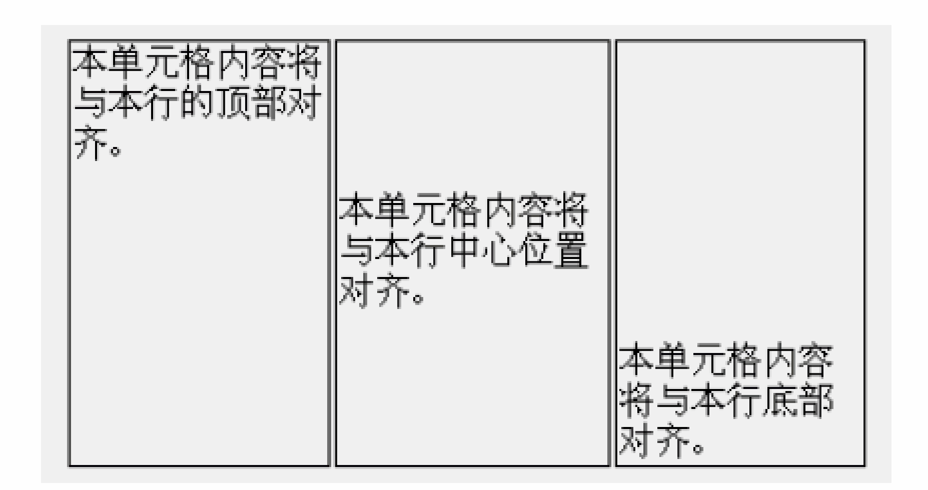


图 13-21　设置 vertical-align 控制垂直对齐方式



13.6 表格实战

13.6.1 斑马线效果

所谓斑马线效果，就是说表格奇数行和偶数行采用不同的样式，在行与行之间形成一种交替变换的效果。只需要给奇数和偶数行指定不同的类名，再分别设定各自的样式即可。

首先准备好如下代码，其中包含表格的(X)HTML 代码和部分样式代码，在此基础上添加斑马线效果：

```
body{
    font-family:Tahoma, "宋体", sans-serif;
    font-size:12px;
}
a img{
    border:0;
}
table{
    border:1px solid #454545;
    border-collapse:collapse;
}
caption{
    padding-bottom:5px;
}
th{
    background:#CCFFFF;
}
td{
    border-top:1px solid #676767;
    border-bottom:1px solid #676767;
    padding:4px;
}
.leftAligned{
    text-align:left;
}
.centerAligned{
    text-align:right;
}
col#teams{
    width:100px;
}
col#scores{
    width:60px;
}
col#more, col#video{
    width:32px;
}
```



```
a{
    text-decoration:none;
}
a:link, a:visited{
    color:#0033CC;
}
a:hover{
    color:#0099FF;
}

<table>
<caption>今日战报</caption>
    <colgroup class="leftAligned">
        <col id="teams" />
        <col id="scores" />
    </colgroup>
    <colgroup class="centerAligned">
        <col id="more" />
        <col id="video" />
    </colgroup>
    <tbody>
        <tr>
            <td>国王 - 76 人</td>
            <td>109 - 99</td>
            <td><a href="more.html">详细</a></td>
            <td><a href="video.html"><img src="images/cam.gif" alt="cam" />
            </a></td>
        </tr>
        ...
        <tr>
            <td>快船 - 灰熊</td>
            <td>98 - 91</td>
            <td><a href="more.html">详细</a></td>
            <td><a href="video.html"><img src="images/cam.gif" alt="cam" />
            </a></td>
        </tr>
    </tbody>
</table>
```

代码省略了中间重复的部分，效果如图 13-22 所示。

今日战报

国王 - 76人	109 - 99	详细	
雄鹿 - 凯尔特人	82 - 104	详细	
猛龙 - 步行者	104 - 93	详细	
魔术 - 山猫	103 - 87	详细	
快船 - 灰熊	98 - 91	详细	

图 13-22　表格初步样式效果

现在分别给奇数和偶数行指定类名，在 tr 元素分别设置类名 odd 和 even。然后添加如下 CSS 规则：

```
tr.odd td{
    background:#AEDEFD;
}
tr.even td{
    background:#DEF2FE;
}
```

以上代码分别给奇数行和偶数行指定不同的背景色，效果如图 13-23 所示，斑马线效果就做好了。

今日战报

国王 - 76人	109 - 99	详细	
雄鹿 - 凯尔特人	82 - 104	详细	
猛龙 - 步行者	104 - 93	详细	
魔术 - 山猫	103 - 87	详细	
快船 - 灰熊	98 - 91	详细	

图 13-23　斑马线效果

13.6.2　日历

日历作为一种工具，可帮助我们安排各种事务，Web 页面中也经常出现日历，比如在线办公系统可以使用日历安排工作内容、博客中的日历可以让我们迅速查看发表于其他时间的文章、某赛事网站提供日历以便我们查看每一天的赛程。作为设计者，我们可考虑使用 table 元素制作日历。

首先创建用于表示日历的(X)HTML 元素，设定 caption 元素和日历的标题行，并将表格按列划分成组：

```
<table>
   <caption>2007 年 11 月</caption>
   <colgroup span="1" class="weekend"></colgroup>
   <colgroup span="5" class="weekday"></colgroup>
   <colgroup span="1" class="weekend"></colgroup>
   <tr id="days">
       <th>日 SUN</th>
       <th>一 MON</th>
       <th>二 TUE</th>
       <th>三 WED</th>
       <th>四 THU</th>
       <th>五 FRI</th>
       <th>六 SAT</th>
    </tr>
```



接下来创建表示日期的单元格，我们以 11 月 1 日为例，之前不包含本月日期的单元格内不写任何代码：

```
    <tr>
        <td></td>
        <td></td>
        <td></td>
        <td></td>
        <td>
            <div class="date">
                <a href="1.html">1</a>
                <span>小光节</span>
            </div>
            <div class="memo">
            </div>
        </td>
```

td 元素内包含两个 div 元素，分别用于显示日期和备忘信息，其他单元格均照此创建。余下的代码为(中间省略了重复的代码)：

```
        <td>
            <div class="date">
                <a href="2.html">2</a><span>廿三</span>
            </div>
            <div class="memo"></div>
        </td>
        <td>
            <div class="date">
                <a href="3.html">3</a><span>廿四</span>
            </div>
            <div class="memo"></div>
        </td>
    </tr>
    <!--第一行结束-->
    <tr>
    ...
        <td>
            <div class="date">
                <a href="30.html">30</a><span>廿一</span>
            </div>
            <div class="memo"></div>
        </td>
        <td></td>
    </tr>
</table>
```

效果如图 13-24 所示。



2007年11月

日 SUN	一 MON	二 TUE	三 WED	四 THU	五 FRI	六 SAT
				1小光节	2廿三	3廿四
4廿五	5廿六	6廿七	7廿八	8立冬	9三十	10十月
11大光节	12初三	13初四	14初五	15初六	16初七	17初八
18初九	19初十	20十一	21十二	22十三	23小雪	24十五
25十六	26十七	27十八	28十九	29感恩节	30廿一	

图 13-24　未添加 CSS 的日历

(X)HTML 代码已经准备好，可以添加样式了。设定表格宽度为 777 个像素，边框模式选为 collapsed，宽度计算方式为 fixed，再添加边框、设定字体大小：

```
table{
    width:777px;
    border-collapse:collapse;
    font-size:12px;
    border:1px solid gray;
    table-layout:fixed;
}
```

下面设置 caption 元素，增大字号，更改字体并设置一定量的底部填充，避免标题和表格之间距离过近：

```
caption{
    font-size:20px;
    font-family:Georgia, sans-serif;
    padding-bottom:6px;
}
```

接下来设定 th 和 td 元素样式。我们给表头指定了背景色，让它与其余单元格区分开来。td 的高度为 110px，其中内容采取顶部对齐：

```
th{
    font-family:Tahoma, "宋体", sans-serif;
    background:#FFF49B;
    padding:3px;
}
td{
    height:110px;
    vertical-align:top;
    padding:0 0 0 3px;
    border:1px solid gray;
}
```

周末和平日要做区分，通过 colgroup 元素选择出表示周末的列，并设置其背景色：

```
colgroup.weekend{
    background:#FFE1DE;
}
```

现在确定每个单元格内公历和农历日期该如何显示。由(X)HTML 代码可知，公历日期放在链接中，表明可以点击查看当天安排的详细情况，而农历日期只起辅助作用，因此应把公历



日期的文字放大并设置农历日期文字颜色为灰色。具体样式代码如下：

```
div.date a{
    width:28px;
    display:block;
    float:left;
    font-size:22px;
    font-family:Georgia, sans-serif;
    text-decoration:none;
}
div.date span{
    color:gray;
    float:left;
    padding-top:8px;
}
```

样式添加得差不多了，可以预览一下效果，如图 13-25 所示。

2007年11月

日 SUN	一 MON	二 TUE	三 WED	四 THU	五 FRI	六 SAT
				1 小光节	2 廿三	3 廿四
4 廿五	5 廿六	6 廿七	7 廿八	8 立冬	9 三十	10 十月
11 大光节	12 初三	13 初四	14 初五	15 初六	16 初七	17 初八
18 初九	19 初十	20 十一	21 十二	22 十三	23 小雪	24 十五
25 十六	26 十七	27 十八	28 十九	29 感恩节	30 廿一	

图 13-25　添加样式后的日历

日历还应该有突出显示当天日期的功能，假如今天是 11 月 8 日，我们给 8 日那一格的 td 增加一个 id 属性，属性值为 today。最后在一些单元格中填写一些内容，这些内容放在类名为 memo 的 div 元素中，再给该 div 添加样式，需要添加和修改的代码如下：

```
div.memo{
    clear:both;
    margin-top:5px;
}
```

```
td#today{
    border:2px solid black;
}

...
        <td>
            <div class="date">
                <a href="14.html">14</a><span>初五</span>
            </div>
            <div class="memo">朋友生日</div>
        </td>
    ...
        <td>
            <div class="date">
                <a href="30.html">30</a><span>廿一</span>
            </div>
            <div class="memo">18:00 同学聚会</div>
        </td>
        <td>
        </td>
    </tr>
</table>
```

最终效果如图 13-26 所示。

2007年11月

日 SUN	一 MON	二 TUE	三 WED	四 THU	五 FRI	六 SAT
				1 小光节	2 廿三 汇报项目进展情况	3 廿四
4 廿五	5 廿六	6 廿七 同学生日	7 廿八	8 立冬	9 三十	10 十月
11 大光节	12 初三	13 初四	14 初五 朋友生日	15 初六	16 初七	17 初八
18 初九 采购日常用品	19 初十	20 十一	21 十二	22 十三	23 小雪	24 十五
25 十六	26 十七	27 十八	28 十九	29 感恩节	30 廿一 18:00 同学聚会	

图 13-26　日历最终效果

读者可继续完善此日历，也可根据需要修改 CSS 代码。另一种常见的日历是微缩型的，尺寸较小，只有简单的日期显示(见图 13-27)，有兴趣的读者可自行编写 CSS 代码实现。



日	一	二	三	四	五	六
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	

图 13-27　微缩型日历

注意： table-layout 属性设为 fixed 后，只有表格宽度是固定不变的。如果单元格内文字内容过多，表格会通过增加高度来容纳多余的内容，即使在 table 或 td 上设置了固定高度也如此。一种在 Firefox 浏览器中可行的解决办法是增加一个 tbody 元素，给该元素设置固定高度并将属性 overflow 设为 hidden，但它在 IE 中并不起作用。通用的办法是在单元格内再添加一个 div 元素，用来对多余的内容进行剪裁。

13.7　小　　结

本章讲解了如何正确使用表格元素，并讲解了与表格相关的各种(X)HTML 标记和常用的样式属性。

表格一般用来存放那些适于表格化显示的数据，而不是作为一种布局工具。

与表格相关的(X)HTML 元素包括 table、colgroup、col、tr、th、td、caption、thead、tbody 和 tfoot。

几乎任何 CSS 属性都可以应用于表格元素，其中边框属性比较重要，CSS 提供了两个边框模式：分离模式和塌陷模式。可通过 border-collapse 属性来设置。

表格宽度计算方式分为固定宽度方式和自动宽度方式，采用前者的表格宽度固定，不论 margin、padding 和 border 的宽度是多少，表格总体宽度值不变；采用后者的表格宽度依据单元格内容的多少而定，宽度的计算速度比前者慢。

通过给 colgroup 和 col 元素添加样式可实现按列控制表格样式。通过调整单元格的 text-align 和 vertical-align 属性可以改变单元格中内容水平和垂直方向的对齐方式。

下一章将介绍如何利用 CSS 控制(X)HTML 中表单元素的样式。

第 14 章　表　　单

表单使用户与 Web 页面进行交互成为可能，通过各种表单元素，用户可以方便地向 Web 服务器提交数据，服务器会根据这些数据完成不同的功能并通过页面表现出来。本章将介绍 (X)HTML 中表单元素的使用以及如何提高它们的易用性，如何使用 CSS 美化表单以及如何对其进行组织、安排。最后将在实例部分介绍一个用户注册表单的设计过程。

本章主要内容

- 基本表单元素
- 如何提高表单元素的访问性
- 如何给表单元素添加样式

14.1　表 单 概 述

如今的 Web 站点不再仅仅提供一些静态信息了，它们更像是应用程序，可以与用户交互，从而完成各种各样的任务。链接使用户可做出选择，然而表单可收集用户信息，使用户可以提交更复杂的信息，进行更复杂的交互。通过表单，用户可以从网上订购商品、搜索文献、在论坛里发表文章、写博客、创建网络相册。对具有交互功能的网站而言，表单是必不可少的。图 14-1 展示的是中国雅虎的用户注册表单。



图 14-1　中国雅虎的用户注册表单

14.1.1　表单元素

本小节会介绍一些表单元素的基本知识，对表单相当熟悉的读者可以跳过这部分内容，阅读下一小节。

表单元素是指(X)HTML 中的 form 元素，该元素中通常会包含一些具有交互功能的(X)HTML 元素，比如文本框、复选框等。这些元素也经常被统称为表单元素，或称为表单控件。

1．输入控件

表示输入控件的(X)HTML 元素是 input，通过设定其 type 属性可实现多种类型的控件：单行文本框、密码文本框、单选框、复选框和提交按钮等。type 属性的取值和对应的控件类型如表 14-1 所示。

表 14-1　input 元素的 type 属性值及对应控件

属 性 值	控 件 说 明
text	单行文本框，可设置 maxlength 属性限制文本框字符的个数
password	密码框，外观同单行文本框，但输入的字符显示为星号或点号
radio	单选框，name 属性相同的多个单选框之中只能有一个处于选中状态
checkbox	复选框
submit	提交按钮，单击此按钮后浏览器会将 form 元素中的所有表单信息提交到服务器端
reset	重置按钮，单击此按钮后，表单信息会被全部清空
button	一般按钮，按钮的行为可由脚本语言定义

input 元素是自我结束(Self-closed)的，和 br 元素一样，不需要结束标记，比如：

```
<input type="text" maxlength="10" name="username" id="username" />
```

2．下拉菜单

表示下拉菜单的(X)HTML 元素是 select，菜单中的每一个选项由 option 元素表示。用户可以从下拉菜单的多个选项中进行选择。下拉菜单平时只会显示其中一项的内容，其余选项需要用户单击下拉按钮才可见。下面是一个下拉菜单的示例，它提供了三个选项：

```
<select name="genre" id="genre">
    <option value="pop">流行</option>
    <option value="classical">古典</option>
    <option value="jazz">爵士</option>
</select>
```

select 元素还有个 size 属性，其默认值为 1，即只显示一个选项，如果给该值设定一个大于 1 的数，则 select 会按照此值显示选项，用户单击上下滚动按钮可看到其他选项。若 select 元素的 multiple 属性为“multiple”时，用户可同时选择多个选项。

3. 多行文本

多行文本是由 textarea 元素表示的，通过 cols 和 rows 属性可设置多行文本框的宽度和高度，单位是字符。如果用户输入的内容超出其可见区域，则多行文本框会自动出现一个滚动条。注意多行文本元素不是自我结束的，需要有结束标记。

4. 辅助元素

辅助元素不是用来接收用户信息的，它们的作用在于实现结构化表单的显示、添加语义化说明、提高表单元素的可访问性和用户的交互体验。

(1) fieldset 元素可将多个 input 元素划分成组，从逻辑上划分为不同的区域。fieldset 元素在浏览器中显示为矩形框。

(2) legend 元素用来描述其父元素 fieldset 的内容，通常浏览器会在 fieldset 矩形框上的适当位置显示 legend 元素中包含的字符。

(3) label 元素和 input 绑定在一起(让 label 元素的 for 属性和 input 元素的 id 属性一致，或者把 input 元素放在 label 元素中)，当用户点击 label 元素时，相应的 input 元素也会作出相应，另外，使用屏幕阅读器的用户可以方便地了解输入框要求输入的是什么内容。

除了以上 3 个元素外，表单元素的 tabindex 和 accesskey 属性也可用来提高表单的可访问性。tabindex 属性可以让用户使用制表键(Tab)切换不同输入控件的焦点。accesskey 属性可设置快捷键，当用户敲击键盘上相应的字符时，控件就会获得输入焦点。

注意：在 IE 浏览器中，如果只将 input 元素放置在 label 元素中，二者并不能实现绑定，必须将 label 元素的 for 属性设置为 input 元素的 id 值。

14.1.2 表单元素的显示

表单在不同浏览器中显示效果差异很大，在 Windows XP 和 Windows Vista 系统中，表单元素的显示还和系统使用的视觉主题有关系。

下面这段(X)HTML 代码包含了常见的表单元素：

```
<form id="myForm">
   <fieldset id="basicInfo">
   <legend>基本信息</legend>
   <label for="username">用户名</label>
   <input type="text" id="username" maxlength="16" />
   <label for="password">密码</label>
   <input type="password" id="password" maxlength="16" />
   <label for="passwordConfirm">确认密码</label>
   <input type="password" id="passwordConfirm" maxlength="16" />
   </fieldset>
   <fieldset id="additionalInfo">
   <legend>附加信息</legend>
   <span>性别</span>
```



```
    <label for="male">
    <input type="radio" name="gender" value="male" id="male" />男
    </label>
    <label for="female">
    <input type="radio" name="gender" value="female" id="female" />女
    </label>
    <label for="selfDescription">
    个人介绍<textarea id="selfDescription" cols="20" rows="5"></textarea>
    </label>
    <label for="vocation">
    从事行业
          <select id="vocation">
             <option value="computer">计算机</option>
             <option value="construction">建筑</option>
             <option value="finance">金融</option>
             <option value="other">其他</option>
          </select>
    </label>
    </fieldset>
    <fieldset id="operation">
    <input type="submit" value="提交" id="submit" />
       <input type="reset" value="重置" id="reset" />
    </fieldset>
</form>
```

图 14-2 至图 14-7 为不同浏览器在不同系统环境下默认的表单显示效果。



图 14-2　IE6 在 Windows XP(经典主题)中的显示效果

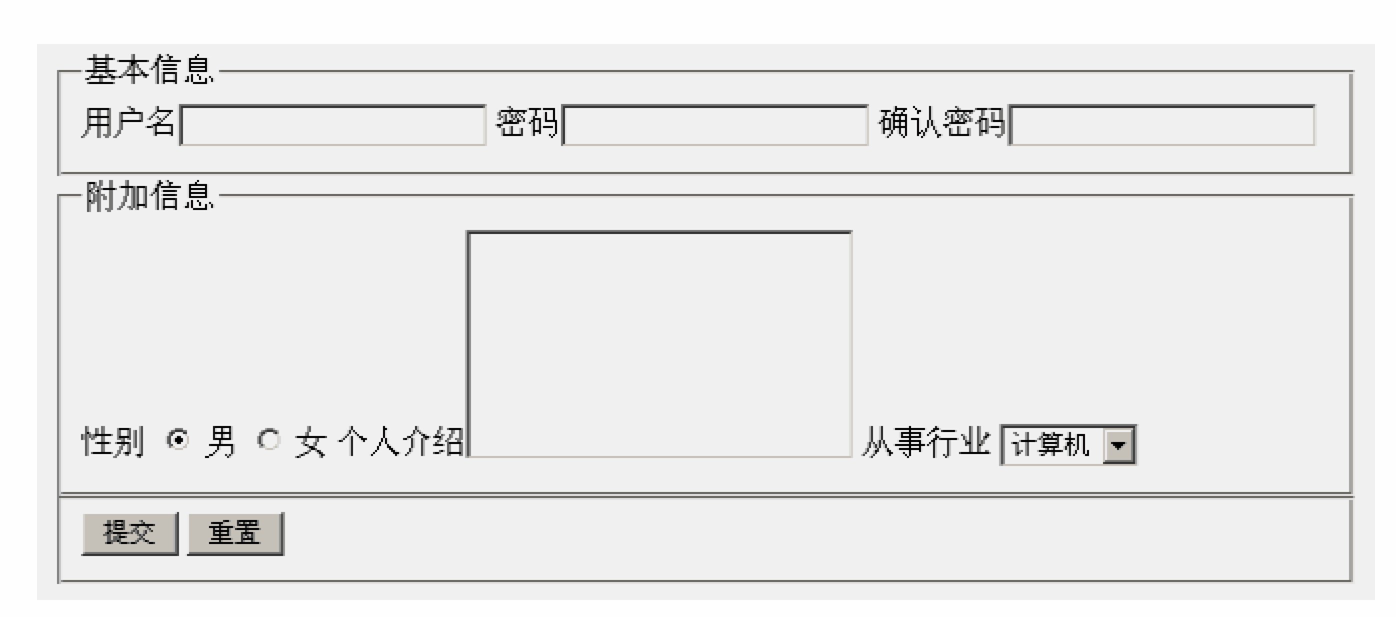


图 14-3　Firefox 在 Windows XP(经典主题)中的显示效果



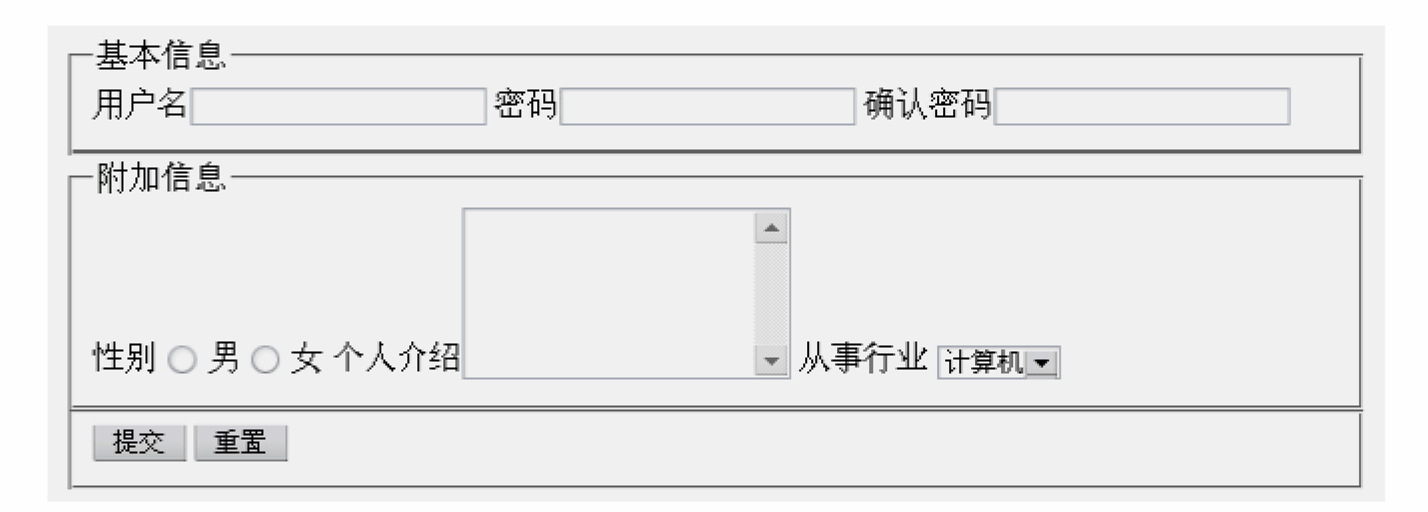


图 14-4　Opera 在 Windows XP 中的显示效果



图 14-5　IE 在 Windows XP(开启主题效果)中的显示效果



图 14-6　Firefox 在 Windows XP(开启主题效果)中的显示效果



图 14-7　IE7 在 Windows Vista 中的显示效果

14.2　添加样式

本节将介绍如何利用适当的 CSS 样式属性来改变浏览器默认的简陋的样式。然后介绍一些常用的组织表单元素的方式，即表单的布局问题。



14.2.1　盒模型相关属性

1. border 属性

盒模型相关属性经常用于文本输入框，IE 默认情况下产生一种立体效果的输入框。添加边框属性就能改变这种默认的样式效果，代码如下：

```
input#input1{
    border:1px solid red;
}
input#input2{
    border:2px dotted #3452DE;
}
input#input3{
    border:none;
}

<input id="input1" type="text" value="文本框" />
<input id="input2" type="text" value="文本框" />
<input id="input3" type="text" value="文本框" />
```

效果如图 14-8 所示。

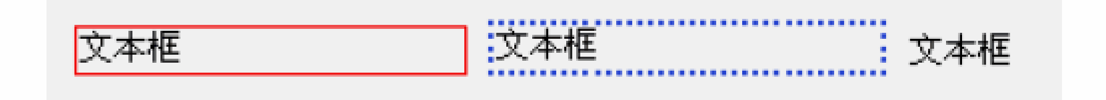


图 14-8　添加 border 属性后的效果

2. padding 属性

padding 属性可以增加文本框中的文字和边框的距离，比如我们给前两个文本输入框添加 padding 属性：

```
input#input1{
    border:1px solid red;
    padding:2px;
}
input#input2{
    border:2px dotted #3452DE;
    padding:4px 8px;
}
```

效果如图 14-9 所示。

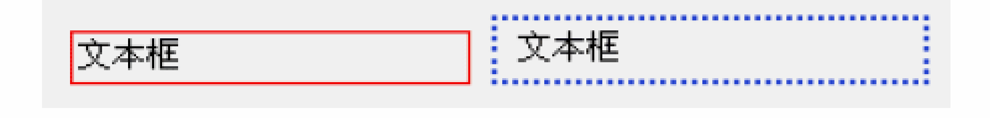


图 14-9　添加 padding 属性后的效果

提示：在 IE 浏览器中，input 元素中的中文文字距上下边框之间的填充并不一致，上填充略小于下填充。如果想让上下填充一致，需要在样式代码中给上下填充属性设置不同的属性值。

3. 宽度和高度

input 的 size 属性可设定该元素的宽度，单位为字符，但是使用 CSS 的 width 和 height 属性可以更精确地控制 input 元素的尺寸。请看如下代码：

```
input#input1{
    border:1px solid red;
    padding:2px;
    height:20px;
    width:100px;
}
input#input2{
    border:2px dotted #3452DE;
    padding:4px 8px;
    height:10px;
    width:6em;
}
```

效果如图 14-10 所示，注意文本框的总宽度和高度包含填充和边框。

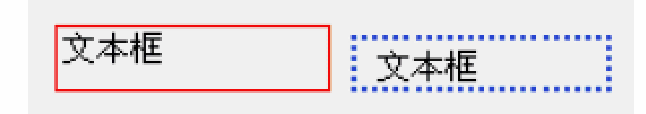


图 14-10　对输入框添加 width 和 height 属性

14.2.2　文字相关属性

表单元素中的文本和其他元素中的文本一样，也可被指定各种不同的文本样式，下面的示例分别给单行文本框、提交按钮和多行文本框指定文字相关的样式：

```
input#searchContent{
    font:italic 12px "Times New Roman", "宋体", serif;
    color:blue;
}
input#submit{
    font:bold 14px "黑体", sans-serif;
    text-decoration:underline;
}
textarea{
    font-size:1.5em;
    line-height:2em;
    text-indent:2em;
}

<input type="text" name="searchContent" id="searchContent" value="[请输入搜索
```



```
内容]"/>
<input type="submit" id="submit" value="搜索" />
<textarea rows="5" cols="10">这里是多行文本输入框</textarea>
```

效果如图 14-11 所示。

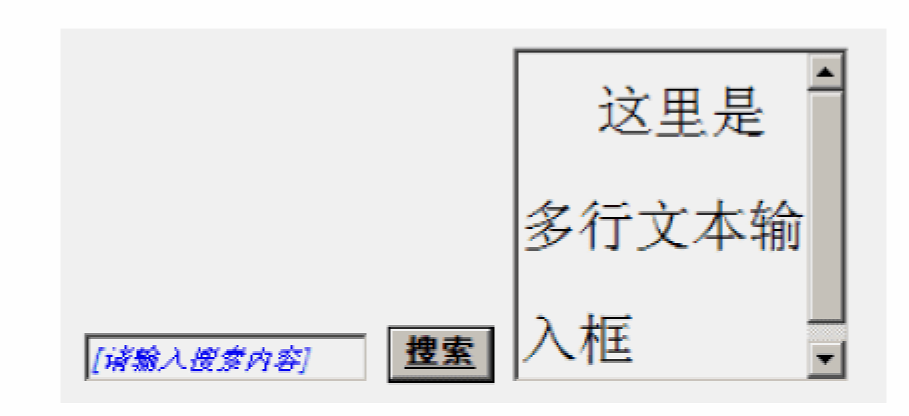


图 14-11　给表单元素指定文本相关的样式

14.2.3　背景和图片

background 属性也可用于表单元素，但是各个浏览器对于添加了背景属性的表单的显示也不尽相同。图 14-12 中由左至右分别为 IE、Firefox 和 Opera 在 Windows XP 下的显示效果。

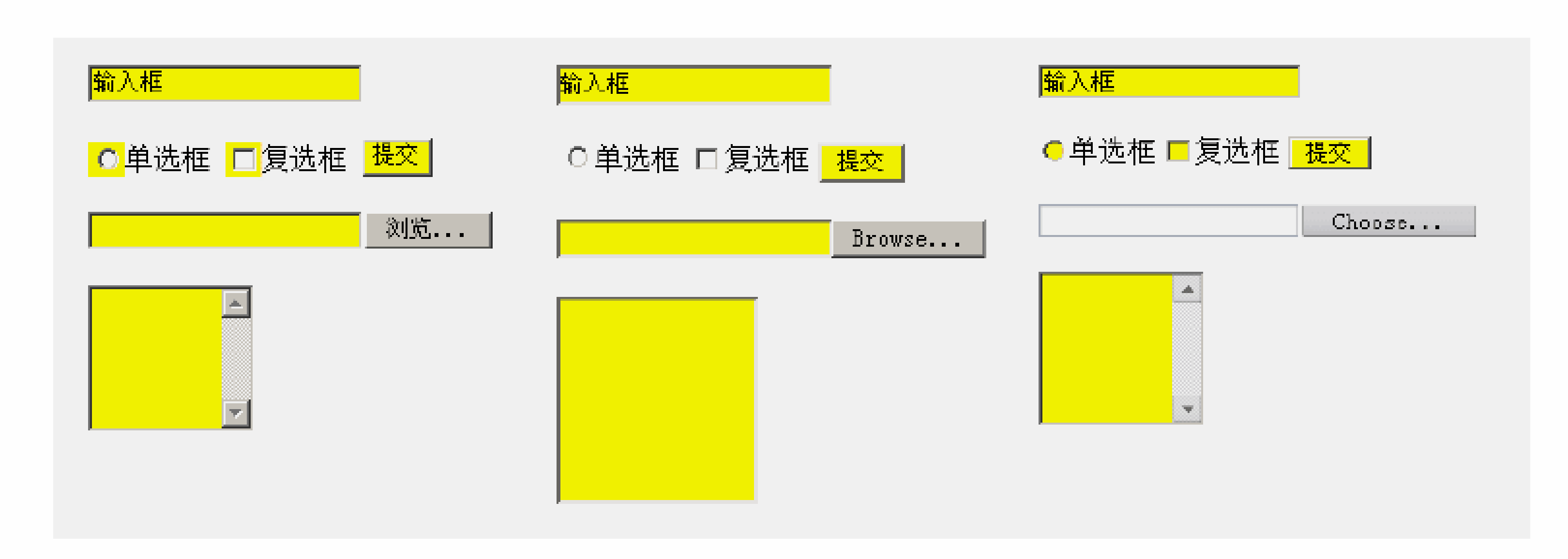


图 14-12　不同浏览器对背景色属性的显示效果(由左至右依次为 IE、Firefox 和 Opera)

现在，在表单含有背景色的基础上添加背景图片(见图 14-13)，图片重复和位置属性取默认值，则会产生如图 14-14 所示的效果。

Hello,CSS!

图 14-13　背景图片



图 14-14　不同浏览器对背景图片属性的显示效果(由左至右依次为 IE、Firefox 和 Opera)



在这 3 款浏览器中，只有单行文本输入框和多行文本输入框效果完全一致，如果只设置背景色或背景图片，则提交按钮的效果也一致(重置按钮、普通按钮也如此)。因此在给表单元素添加样式时需要考虑不同浏览器的显示效果。

14.2.4 表单元素的布局

1. 表格布局

读者可能会有疑问：不是说过不推荐使用表格布局吗？

实际上使用 CSS 对表单元素布局比较困难，需要花费大量的时间和精力才能做出比较满意的效果。有兴趣的读者可以亲自试一试。复杂的表单符合表格化数据的形式(想想你填写过的申请表、体检表等，它们都是放置在表格之中的)，因此使用表格来安排表单元素是合理的，也是最佳选择。

本章的实战部分将会详细介绍如何使用表格对表单元素进行布局。

2. 段落布局

段落布局是将表单元素放在 p 元素内，这种方式适合相对比较简单的表单。使用段落布局的(X)HTML 代码如下：

```
<form action="" method="post" id="messageForm">
    <fieldset>
        <legend>留言簿</legend>
        <p>
            <label for="name">姓名</label><br />
            <input type="text" name="name" id="name" tabindex="1" />
        </p>
        <p>
            <label for="email">电邮</label><br />
            <input type="text" name="email" id="email" tabindex="2" />
        </p>
        <p>
            <label for="message">留言内容</label><br />
            <textarea name="message" id="message" rows="8" cols="30" tabindex="3">
            </textarea>
        </p>
        <p>
            <label for="subject">你是如何知道本网站的</label><br />
            <select name="subject" id="subject" tabindex="4">
                <option value="">--请选择--</option>
                <option value="Option1">友情链接</option>
                <option value="Option2">朋友介绍</option>
                <option value="Option3">搜索引擎</option>
            </select>
        </p>
        <p>
            <label for="receive">我愿意通过邮件接收本网站的资讯</label>
```



```
            <input type="checkbox" name="receive" id="receive" value="n"
             tabindex="5"/>
        </p>
    </fieldset>
    <input type="submit" value="提交" tabindex="6" />
</form>
```

添加如下样式信息：

```
form {
    font-size:13px;
    width:340px;
}
fieldset {
    padding:3px;
    border:1px solid #333;
}
legend {
    background-color: #DDDDDD;
    padding:5px;
    font-size:16px;
    font-weight:bold;
}
input, textarea{
    font-family: Arial, "宋体", sans-serif;
}
#name, #email{
    width:200px;
    border:1px solid #999;
}
#message{
    width:300px;
    border:1px solid #999;
}
#submit{
    margin-top:10px;
}
```

以上样式设定了表单的整体宽度，表单内元素文字的大小，辅助元素的填充、背景，最后设置了输入框的宽度和边框样式，效果如图 14-15 所示。

如果还感觉它有些单调，可以给每个 p 元素增加背景色，并修改默认的 margin 属性：

```
form p{
    background:#EDFBFF;
    padding:6px;
    margin:2px 0;
}
```

效果如图 14-16 所示。



图 14-15　添加样式后的表单



图 14-16　最终效果

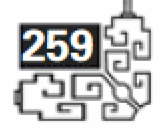
14.3　表 单 实 战

在本节的实战中，我们将完成一个网站注册页面的制作，同时介绍如何使用表格来安排表单元素。通常表单元素会放在一个两列多行的表格中，第一列是 label 元素，第二列是各种表单控件。

首先准备好(X)HTML 代码：

```
<form action="" name="signup" id="signup">
<table>
    <col id="colHeader" />
    <col id="colInput" />
```

```
<thead>
    <tr><th colspan="2">注册</th></tr>
</thead>
<tbody>
<tr class="sectionHeader"><th colspan="2">以下内容为必填信息</th></tr>
<tr>
    <th><label for="username">账号</label></th>
    <td>
    <input type="text" id="username" name="username" size="16"
      tabindex="1" />
    </td>
</tr>
<tr>
<th><label for="password">密码</label></th>
    <td>
    <input type="password" id="password" name="password" size="16"
      tabindex="2" />
    </td>
</tr>
<tr>
    <th><label for="passwordConfirm">密码确认</label></th>
    <td>
    <input type="password" id="passwordConfirm" name="passwordConfirm"
      size="16" tabindex="3" />
    </td>
</tr>
<tr>
    <th><label for="email">电子邮件</label></th>
    <td>
    <input type="text" id="email" name="email" size="30" tabindex="4" />
    </td>
</tr>
</tbody>
<tbody>
<tr class="sectionHeader"><th colspan="2">以下内容为选填信息</td></tr>
<tr>
    <th><label for="questionid">安全提问</label></th>
    <td>
        <select id="questionid" name="questionid" tabindex="5">
            <option value="0">无安全提问</option>
            <option value="1">母亲的名字</option>
            <option value="2">爷爷的名字</option>
            <option value="3">您所养的宠物的名字</option>
            <option value="4">您其中一位老师的名字</option>
            <option value="5">您个人计算机的型号</option>
            <option value="6">您最喜欢的餐馆名称</option>
            <option value="7">驾驶执照的最后四位数字</option>
        </select> 启用安全提问后，登录时需填入相应项目
    </td>
```



```
    </tr>
    <tr>
        <th><label for="answer">回答</label></th>
        <td>
        <input type="text" id="answer" name="answer" size="25" tabindex="6" />
        </td>
    </tr>
    <tr>
        <th><label>性别</label></th>
        <td>
            <label for="male">
            <input type="radio" name="gender" value="1" id="male"
                tabindex="7" />男
            </label>
            <label for="female">
            <input type="radio" name="gender" value="2" id="female"
                tabindex="8" />女
            </label>
            <label for="keepsec">
            <input type="radio" name="gender" value="0" id="keepsec"
                checked="checked" tabindex="9" />保密
            </label>
        </td>
    </tr>
    <tr>
        <th><label for="birthday">生日</label></th>
        <td>
        <input type="text" name="birthday" id="birthday"
          value="0000-00-00" tabindex="10"/>
        </td>
    </tr>
    <tr>
        <th><label for="qq">QQ</label></th>
        <td><input type="text" name="qq" id="qq" tabindex="11" /></td>
    </tr>
    <tr>
        <th><label for="msn">MSN</label></th>
        <td><input type="text" name="msn" id="msn" tabindex="12" /></td>
    </tr>
    <tr>
        <th><label for="icq">ICQ</label></th>
        <td><input type="text" name="icq" id="icq" tabindex="13" /></td>
    </tr>
    <tr>
        <th><label for="bio">自我介绍</label></th>
        <td>
        <textarea name="bio" id="bio" cols="40" rows="5" tabindex="14">
        </textarea>
        </td>
```



```
    </tr>
    <tr>
        <th></th>
        <td>
        <label for="autoSignIn">自动登录
        <input type="checkbox" name="autoSignIn" id="autoSignIn" tabindex="15" />
        </label>
        </td>
    </tr>
    </tbody>
    <tbody>
    <tr>
        <th></th>
        <td>
        <input type="submit" value="提 交" id="submit" tabindex="16" />
        </td>
    </tr>
    </tbody>
</table>
</form>
```

以上代码采用表格布局表单元素，table 元素放在 form 元素之内。表格按行划分为 4 个逻辑区域(用 1 个 thead 元素和 3 个 tbody 元素区分)，分别表示标题、必填内容、选填内容和提交。表格第一列使用 th 元素，表明它是右侧输入控件的标题，第二列使用普通的 td 元素，存放各种输入控件。好了，用表格布局的内容就这么多，下面该轮到用 CSS 来美化这个注册页面了。

首先添加一些基本的样式，包括字体、颜色、表格的初步样式：

```
*{
    font:12px Tahoma, Arial, "宋体", sans-serif;
    color:#333;
}
form#signup table{

    width:520px;
    table-layout:fixed;
    border-collapse:collapse;
    text-align:left;
    border:1px solid #666;
}
th, td{
    border-bottom:1px solid #666;
    padding:5px;
}
th{
    font-weight:normal;
}
col#colHeader{
    width:100px;
}
```

以下样式用来处理表格的标题和各个逻辑区域的提示信息，给它们添加不同的背景色和文字颜色以示区别。代码如下：

```
thead th{
    background-color:#8AA936;
    color:white;
    border-bottom-color:#425118;
}
tr.sectionHeader th{
    background-color:#FDFFC7;
    color:#999;
}
```

效果如图 14-17 所示。

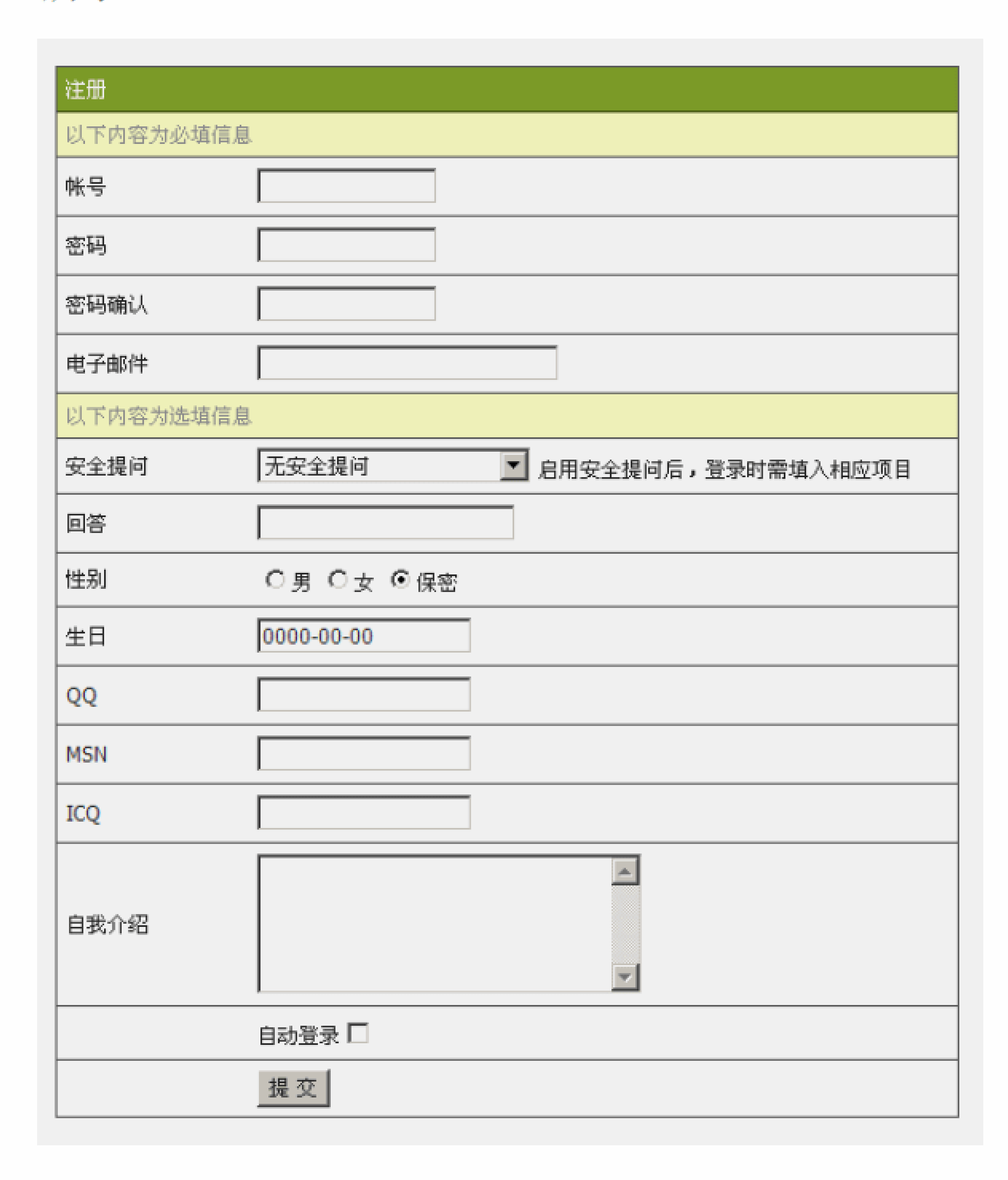


图 14-17　表单初步样式

继续给表单元素添加样式：

```
input, select{
    width:160px;
}
textarea{
    width:260px;
}
input, textarea{
    border-width:1px;
```

```
    border-style:solid;
    border-color:#333 #BBB #BBB #333;
}
input#male, input#female, input#keepsec, input#autoSignIn, input#submit{
    width:auto;
    border:none;
}
```

效果如图 14-18 所示。

图 14-18　给表单元素增加样式

在性别一栏中，我们打算使用图片替换掉原来的文字，因为图片比文字更直观。为此需要给表示“男”、“女”选项的 label 元素添加 id 属性，以便为其增加不同的背景图，同时给文字增加一个 span 元素并通过 display 属性将其隐藏。如果浏览器不支持 CSS 或者用户将其禁用，那么性别选项依然会显示“男”、“女”二字。label 元素的 id 分别为 lblMale 和 lblFemale，修改后的(X)HTML 部分代码如下：

```
<label id="lblMale" for="male"><input type="radio" name="gender" value="1"
id="male" tabindex="7" /><span>男</span></label>
<label id="lblFemale" for="female"><input type="radio" name="gender" value="2"
id="female" tabindex="8" /><span>女</span></label>
```

样式代码如下：

```
label#lblMale span, label#lblFemale span{
    display:none;
```

```
}
label#lblMale{
    padding-right:16px;
    background:url(images/head_male.gif) no-repeat right;
}
label#lblFemale{
    padding-right:16px;
    background:url(images/head_female.gif) no-repeat right;
}
```

最后要处理的就是提交按钮了，我们同样给它增加背景图片，让页面更美观：

```
input#submit{
    background:url(images/btn_bg.gif) no-repeat left top;
    width:65px;
    height:27px;
}
```

最终效果如图 14-19 所示。

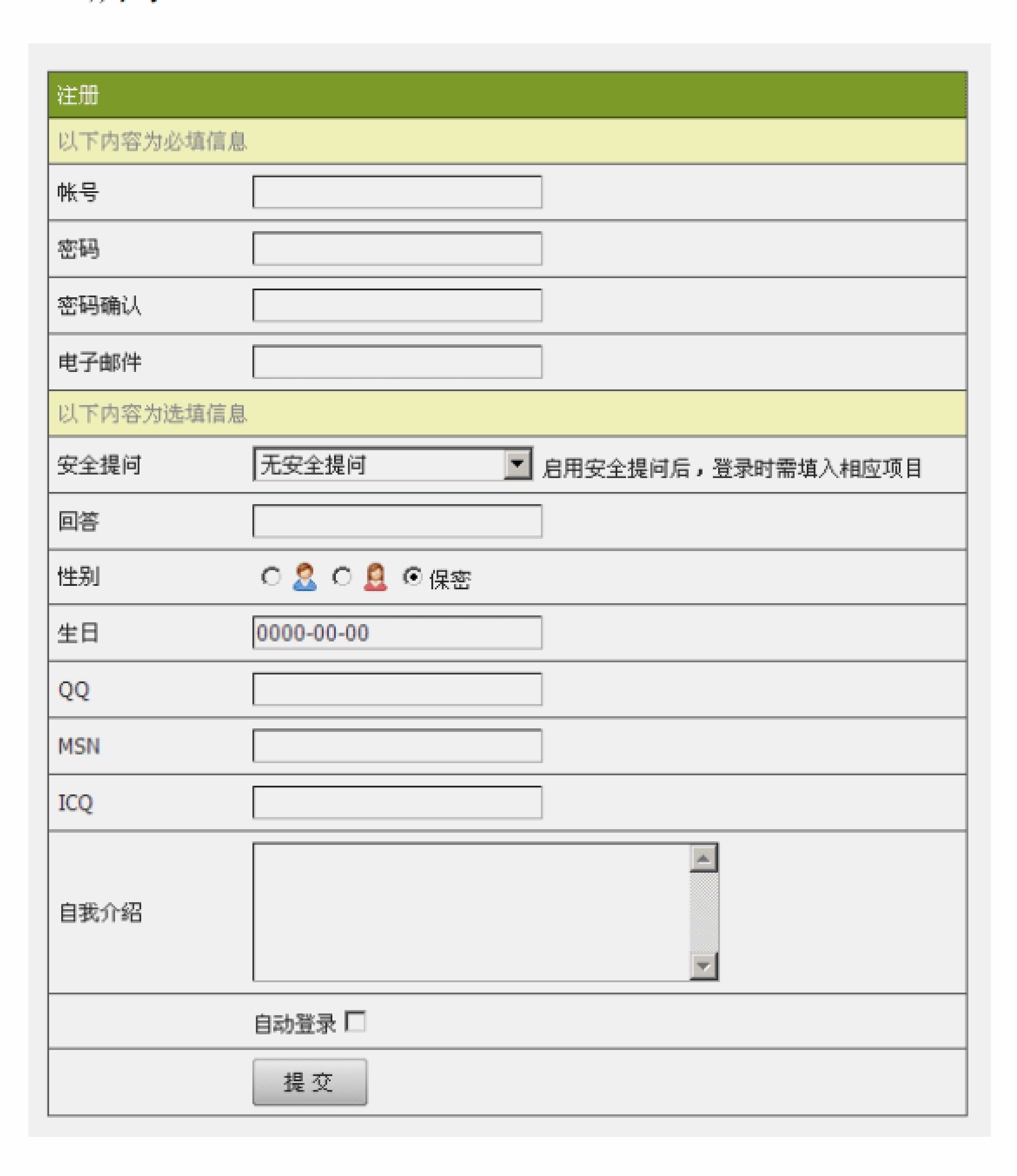


图 14-19　表单最终效果(表示性别的图标截取自腾讯的 TM2008)

这时，如果用 Firefox 浏览器打开此页面，会发现性别一栏中图像顶端有一部分不见了。而在 Opera 中，图像上下各有一小部分也消失了(见图 14-20)。

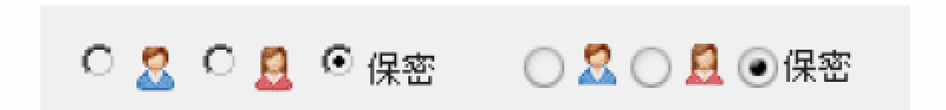


图 14-20　Firefox(左)和 Opera(右)浏览器中的显示效果



这个问题的原因在于，label 元素的高度在这两款浏览器中是随着其中的文字大小变化而定的，若 font-size 属性大于 12px 则不会有此问题(比如 13px，如图 14-21 所示)。

图 14-21　将 font-size 属性设为 13px 后，问题可以得到解决(左：Firefox，右：Opera)

现在要求不改变文字大小，那么可以使用 padding 属性为 label 添加一些填充，把 label 的空间扩大，则可以完整地显示出背景图像。修改之后的 CSS 代码如下：

```
label#lblMale{
    padding:1px 16px 1px 0;
    background:url(images/head male.gif) no-repeat right;
}
label#lblFemale{
    padding:1px 16px 1px 0;
    background:url(images/head female.gif) no-repeat right;
}
```

现在，三款浏览器都能获得比较满意的显示效果了。

14.4　小　结

表单的使用使得用户可以与 Web 页面进行复杂的交互，本章介绍了表单元素的基本知识、如何添加适当的样式属性以及如何对表单元素进行布局。

表单元素包括各种输入控件、菜单和一些辅助性的元素，辅助性元素可以提高表单的易用性，另外表单元素的 tabindex 和 accesskey 属性也能有效地提高表单的可访问性。

在未添加任何样式之前，表单元素在不同平台不同浏览器上的显示差异比较大，虽然通过 CSS 样式可以统一部分样式，但是由于各种浏览器对 CSS 的支持程度不同，要想设计出效果在各浏览器中都完全相同的表单几乎是不可能的。我们能做到的只是尽量统一它们的样式。

盒模型相关的属性、背景和控制文字的属性都可用于表单元素。表单元素可以利用表格或段落来实现布局。

最后，本章实例部分介绍了一个完整的用户注册表单的设计过程。

从下一章开始，将进入本书的第四部分——CSS 布局，第 15 章将介绍与布局紧密相关的浮动与定位技术。

第四篇　CSS 布局技术

第 15 章　浮动与定位

盒模型、浮动和定位是 CSS 中三个非常重要的概念。其中浮动和定位这两个概念是进行 CSS 网页布局的基础，它们决定了文档元素如何布局，以何种方式显示在浏览器的窗口中。

在以前，网页设计人员使用表格来进行网页布局。如果你以前也用表格进行布局，那么抛弃表格并通过 CSS 控制布局可能会使你感到不适应。但是当你完全理解上述概念之后，使用 CSS 布局将是一件十分轻松甚至愉悦的事。

CSS 定位可以帮助设计人员精确地确定(X)HTML 元素在文档中的位置。除了控制元素水平和垂直方向上的位置外，CSS 还能够设定元素的深度，即元素重叠的顺序。

本章主要内容

- 什么是(X)HTML 文档流
- float 属性的使用
- clear 属性的含义
- 如何利用 float 属性进行页面布局
- IE 中的浮动问题以及解决方案
- position 属性和 CSS 定位的类型
- 什么是静态定位、相对定位、绝对定位和固定定位
- left、right、top 和 bottom 属性的作用
- 如何利用定位实现页面布局
- 如何使用 z-index 属性控制元素深度
- visibility 属性和元素的可见性

15.1　float 属性

15.1.1　(X)HTML 文档流

前面说过，(X)HTML 元素可分为两大类：内联元素和块级元素。二者在显示方式上是不同的。

内联元素是在水平方向上一个接一个排列的，元素前后不产生换行。元素间的水平间距可以通过水平方向上的填充、边框和间距来控制。常见的 strong、span 等就属于内联元素。需要注意的是，竖直方向上的填充、边框和间距对控制内联元素的高度是不起作用的。水平方

向的一行将构成一个所谓的“Line Box”，这是一个逻辑上的概念，“Line Box”是一个虚构的矩形区域，包含了组成该行的所有内联元素，其高度等于本行内所有内联元素中高度值最大的那一个。在显示内联元素时，浏览器会自动计算出合适的高度以便将元素内容完全显示出来。尽管浏览器自动计算内联元素高度，我们还是可以通过设置 line-height 属性的值来改变元素的高度。

块级元素是在竖直方向上一个接一个排列的，元素前后均产生换行。竖直方向上元素之间的距离可以用上下边距来控制，注意竖直方向的边距会产生重叠，其间距值取相邻元素中边距值较大的那一个。常见的 p、div 等都是块级元素。这些元素的内容都是以“块”的形式显示在浏览器中的。

内联元素和块级元素各自遵循着不同的显示方式，这就构成了(X)HTML 的文档流(Flow)。文档的内容被比喻为“流”。文档中的元素可以“随波逐流”，也可以脱离“流”，“漂浮”在其上。如果你没有为元素设定附加的 CSS 属性，那么元素将按照它在(X)HTML 代码中出现的顺序一个挨着一个地排列。

请看如下代码：

```
div#box1, div#box2, div#box3{
    width:80px;
    height:80px;
    border:1px solid #000;
}

<div id="box1">块级元素竖直方向依次排开</div>
<strong>内联元素</strong><span>水平方向依次排开</span>
<div id="box2">块级元素竖直方向依次排开</div>
<div id="box3">块级元素竖直方向依次排开</div>
<strong>内联元素</strong><span>水平方向依次排开</span>
<strong>内联元素</strong><span>水平方向依次排开</span>
```

为了可以看清楚效果，我们为三个 div 元素设置了高度、宽度以及边框。图 15-1 展示了代码在浏览器中的效果。

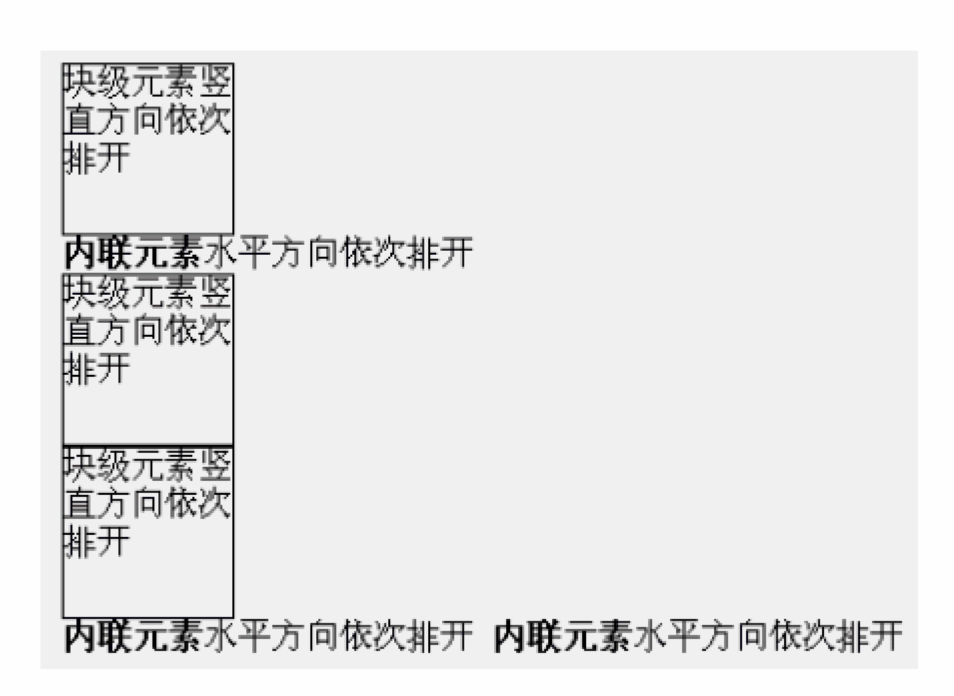


图 15-1　在默认的文档流中，块级元素竖直排列，内联元素水平排列

div 是块级元素，strong 和 span 是内联元素。div 元素的前后都有换行，而 strong 和 span 元素之间以及 span 元素之间不产生换行。注意最后一个 strong 元素与前一个 span 元素之间存



在一个空格，这是由(X)HTML 代码中代码之间的换行符导致的。

15.1.2 float 属性的作用

CSS 的 float 属性可以使元素脱离原始的文档流，使其浮动于原始的文档流之上。浮动元素会向左或向右靠在包含该元素的容器一边，但它的位置还是基于正常的文档。浮动元素不再占用原来的位置，因此位于文档流中的其他元素(即未被设置为浮动的元素)会填补该元素原来的位置，这些元素中的内容会围绕在浮动元素周围。float 属性可用于任何(X)HTML 元素。设置了浮动属性的元素遵循如下规则：

- 浮动元素的边距不会产生重叠现象。
- 只有元素中的内容会受到浮动元素的影响。这就是说背景、边距、边框、填充等盒模型相关属性不受影响。
- 浮动元素被视为块级元素，元素维度由边距、边框、填充、高度和宽度属性决定。

图 15-2 展示了一个向左浮动的元素和其父元素(即容器)的位置关系，图中浅灰色线条表示父元素中的内容。

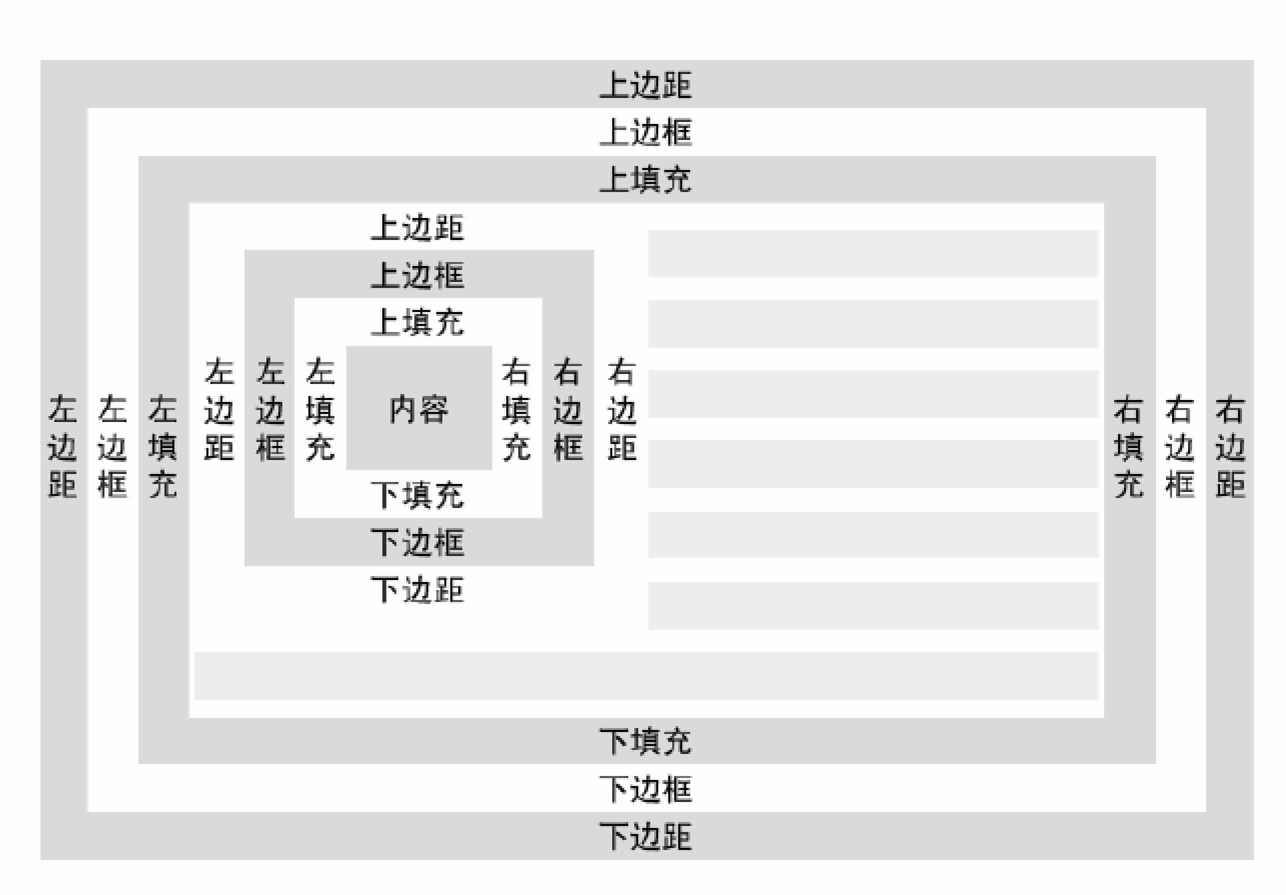


图 15-2　浮动元素的盒模型示意图

float 属性值可以是 left、right 和 none。left 表示元素向左浮动，right 表示元素向右浮动，而 none 表示元素不浮动。

请看示例：

```
body{
    border:1px dashed gray;
    padding:10px;
}
div{
    height:80px;
    width:80px;
    border:1px solid #000;
}
```



```
<div id="box1">box1</div>
<div id="box2">box2</div>
```

如图 15-3 所示为以上代码的效果。div 属于块级元素，在竖直方向上进行排列。

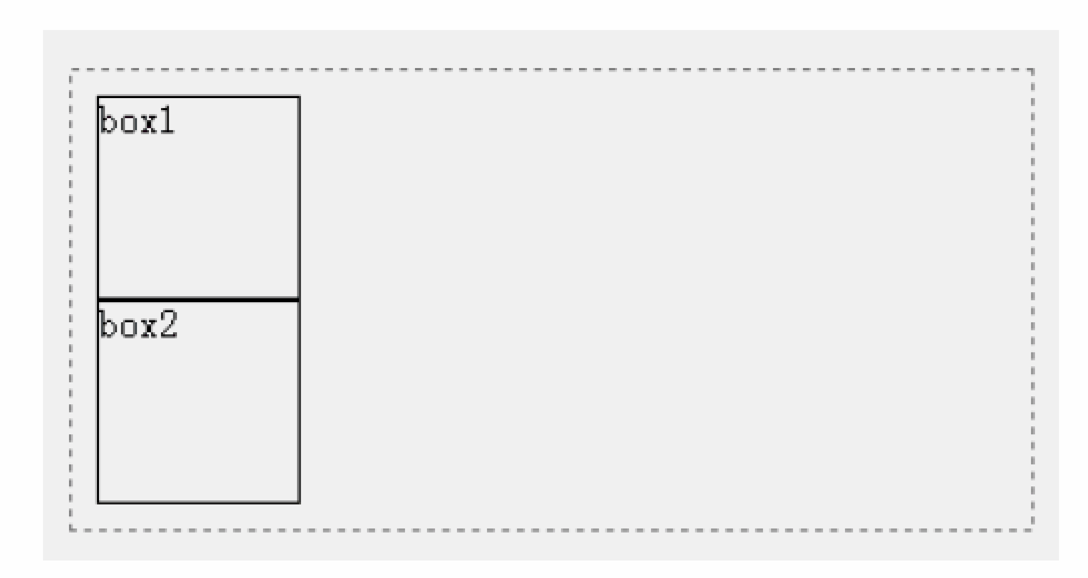


图 15-3　未使用 float 属性时，元素按照默认的方式排列

现在我们给#box1 添加 float 属性，让它向右浮动。

```
div#box1{
    float:right;
}
```

图 15-4 显示了给#box1 添加浮动属性之后的效果，#box1 脱离了原始的文档流向右浮动，直到碰到#box1 的容器，即 body 元素，#box2 替代了原来#box1 的位置。图中的箭头表示了元素在原始文档流中的移动方向。

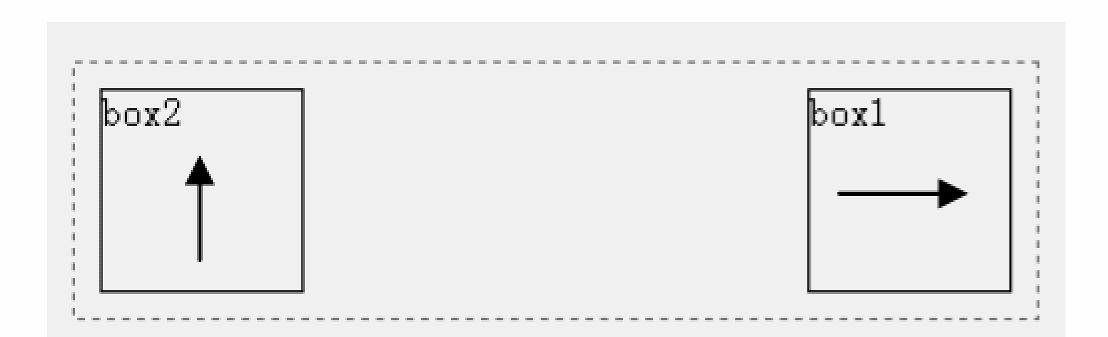


图 15-4　给#box1 添加 float 属性，使其脱离原始文档流向右浮动

现在我们让#box1 向左浮动，同时给两个 div 元素设置不同的背景色，最后增加#box2 的宽度和高度：

```
div#box1{
    float:left;
    background:pink;
}
div#box2{
    background:yellow;
    width:200px;
    height:120px;
}
```

如图 15-5 所示为 Firefox 浏览器中的显示效果。



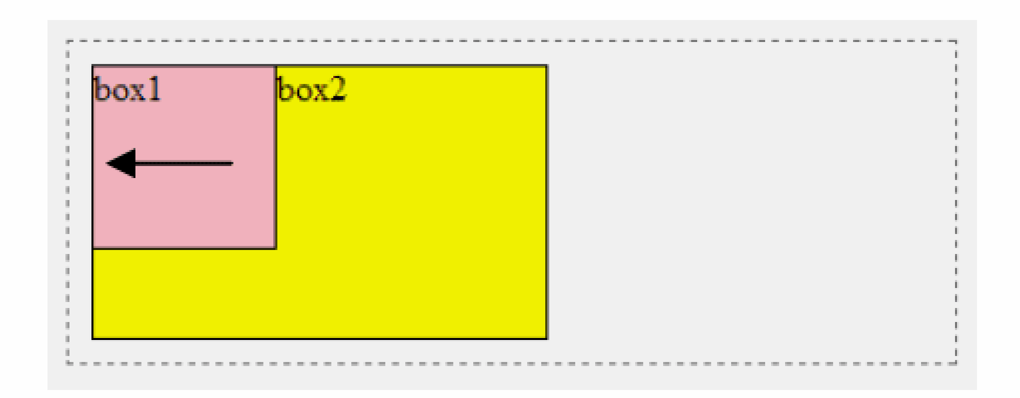


图 15-5　将 box1 向左浮动(Firefox 浏览器)

当#box1 向左浮动后，#box2 占据了#box1 的位置，注意浮动元素只影响其余元素中的内容，因此#box2 中的文字没有被#box1 所覆盖，而是围绕在它的旁边。注意 IE 浏览器并没有严格遵循 CSS 的规范，当元素设定了确定高度或宽度时，它将靠在浮动元素旁边，而不会占据浮动元素的位置。本例的#box2 就靠在了#box1 旁边，如图 15-6 所示。

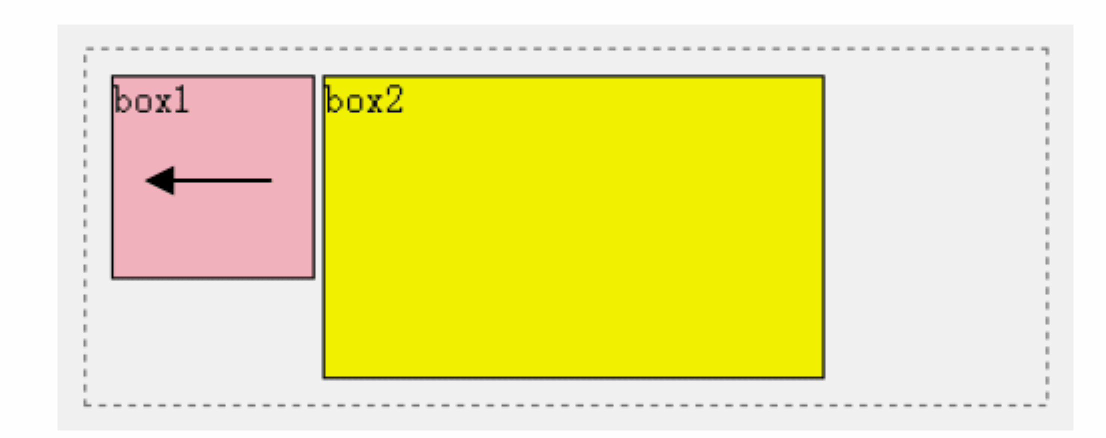


图 15-6　IE 浏览器的显示效果，#box2 并没有占据#box1 的位置，而是挨在#box1 的旁边

如果我们将#box1 和#box2 都向左浮动，那么它们会一个挨着一个浮动在浏览器窗口中：

```
div{
    float:left;
}
```

如图 15-7 所示为 IE 浏览器的显示效果。

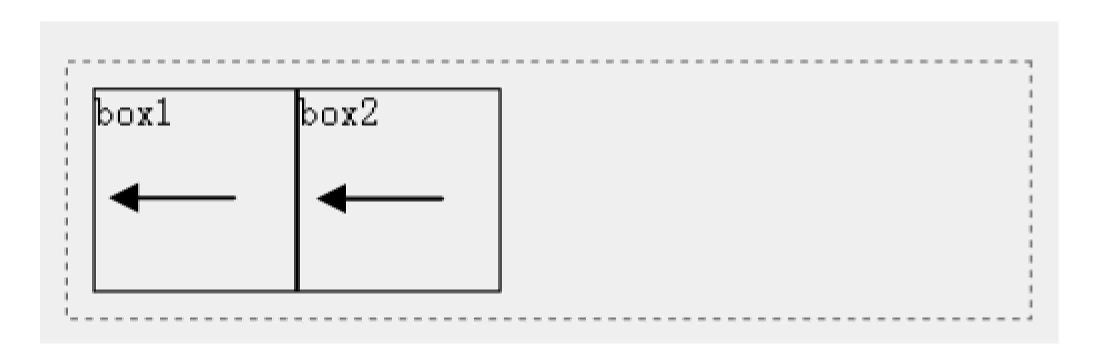


图 15-7　#box1 和#box2 均向左浮动(IE 浏览器)

如图 15-8 所示为 Firefox 浏览器的显示效果，与 IE 不同的是，body 元素并没有随着两个 div 元素的高度而变化。这是因为浮动的 div 元素脱离了原始的文档流，也就不再被 body 元素所包含，现在 body 元素没有包含的元素，高度也就为 0 了。在下一小节我们将介绍如何处理此种情况下父元素的高度问题。

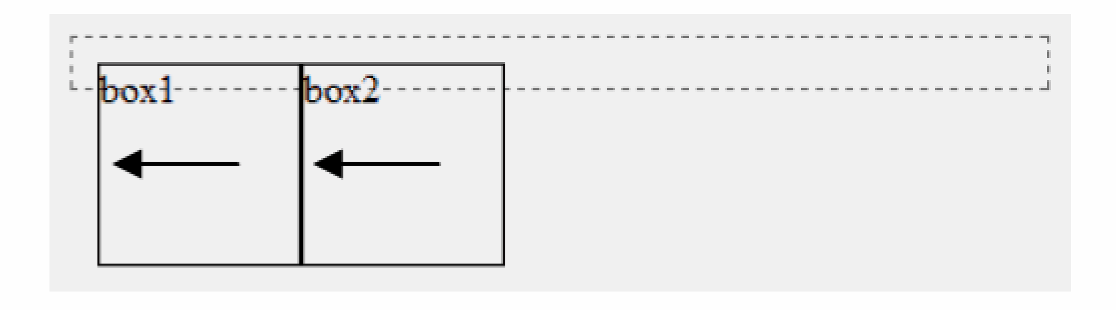


图 15-8　Firefox 浏览器的显示效果，body 元素的高度没有根据 div 元素变化



当 body 元素的宽度能够容纳#box1 和#box2 时，这两个 div 水平方向依次排开。当我们逐步减小浏览器窗口的宽度时，#box2 就会被挤到#box1 的下面(见图 15-9)。这种情况被称作浮动下落(Float Drop)，在使用浮动进行页面布局时尤其要注意避免这种现象发生。

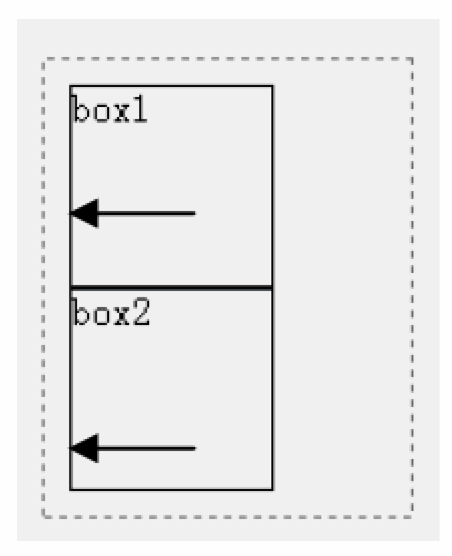


图 15-9　浏览器宽度缩小后#box2 被挤到#box1 的下面

注意：浮动只能向左或向右进行，不会浮动在中间某个位置，读者在刚接触浮动概念时需要注意。

15.2　clear 属性

前面讲过，浮动元素会影响位于正常文档流中的元素内容排列，使内容围绕在浮动元素的周围。在第 11 章中，我们曾经利用 float 属性进行图文混排，使图片向左或向右浮动，这时文字就会围绕在图片的周围。再举个例子：

```
body{
    font-family:"Times New Roman", "宋体", serif;
}
div#wrapper{
    width:540px;
    border:1px solid gray;
    padding:0 10px;
}
p{
    text-indent:2em;
    margin:10px 0;
    line-height:1.3em;
}
.floatLeft{
    float:left;
}
img#flower{
    margin:10px 10px 10px 0;
}

<div id="wrapper">
    <img class="floatLeft" id="flower" src="images/flower.jpg" alt="flower" />
```



```
    <p id="para1">火鹤原产于南美洲热带雨林中，属热带性植物，多年生草本。其花呈腥红色，肉穗状花序向外倾斜，上端黄色，下端白色。整个花像一只伸展的红色手掌，掌心竖起一条弯曲的金黄色肉穗，仿佛又像是鹤的颈项。</p>
    <p id="para2">火鹤又名红掌、花烛、安祖花、烛台花。其英文名称为 anthurium，由前缀 anthos(花)和 oura(尾)组成。火鹤的肉穗花序就如同一条尾巴，因此得名。</p>
</div>
```

以上代码将产生如图 15-10 所示的效果，#para1 和#para2 内的文字都会围绕在浮动的 img 元素的周围。

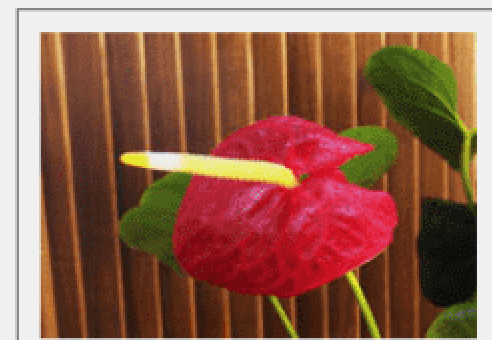


图 15-10　p 元素的文字都会围绕在浮动的 img 元素周围

CSS 的 clear 属性可以禁止这种情况发生，使文字不会环绕在这个浮动元素周围。clear 属性值可以为 left、right、both 和 none。left 表示元素内容不会围绕在向左浮动的元素周围，right 表示元素内容不会围绕在向右浮动的元素周围，both 相当于同时使用 left 和 right，none 则表示元素内容总产生围绕效果。

现在给#para2 添加如下样式规则：

```
p#para2{
    clear:left;
}
```

#para2 中的文字不会围绕在 img 元素周围，它按照块级元素的显示规则，在图像后面另起新的一行显示(见图 15-11)。

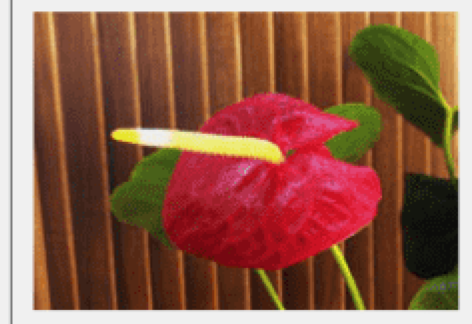


图 15-11　#para2 使用 clear 属性后，其内容不再围绕浮动元素

现在若将#para2 从(X)HTML 文档中除去，则 Firefox 浏览器会产生如图 15-12 所示的效果。

前面说过，容器不会将浮动元素包含在内，因此 div 元素的高度只根据其中的文字内容而定，所以出现了如图 15-12 所示的效果，图像越出了 div 元素的下边界。我们可以使用 clear 属性来解决这个问题。



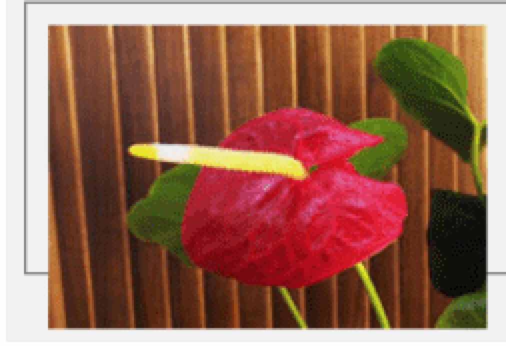


图 15-12　将#para2 去掉后，Firefox 浏览器的显示效果

现在在#para1 后面添加一个 div 元素，设置其 class 属性为 clear：

```
<p id="para1">火鹤原...</p>
<div class="clear"></div>
```

再给.clear 添加如下样式规则：

```
.clear{
    clear:both;
}
```

效果如图 15-13 所示，#wrapper 的下边界能够将图像包含在内了。

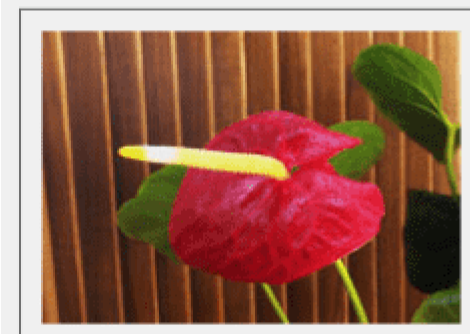


图 15-13　使用 clear 解决容器无法包含浮动元素问题

我们添加了一个空的 div 元素，并给它添加了 clear 属性。图 15-14 展示了添加该元素所起到的作用。

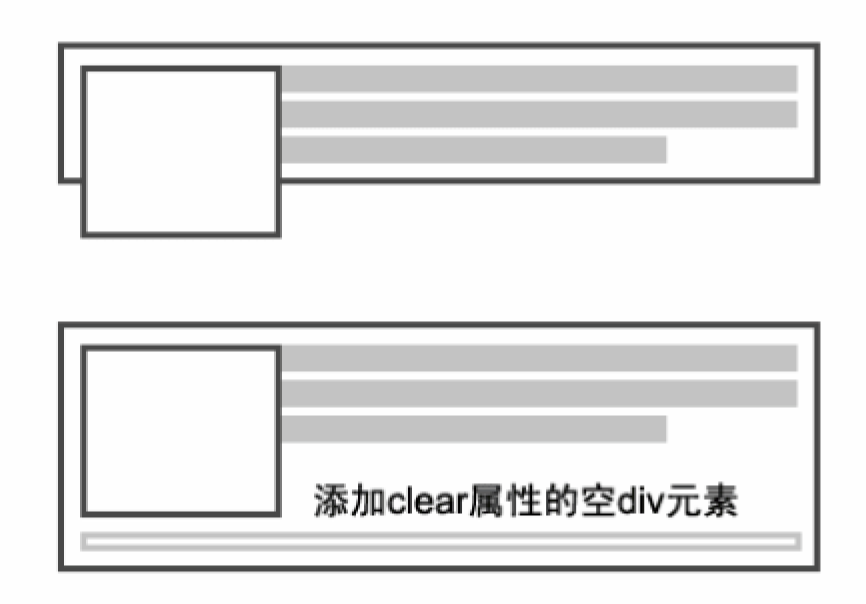


图 15-14　添加 clear 属性的空 div 元素的作用

从图 15-14 看出，这个空元素将#wrapper 的高度强制拉伸至图像的高度。

提示： 严格地讲，第 11 章中介绍的图文混排范例的健壮性不够高，如果最后一段文字内容较少，则会产生类似图 15-12 所示的效果。为了避免这一现象的发生，就要添加额外的元素，并为其设定 clear 属性。



15.3 使用 float 属性布局页面

使用 float 属性除了能进行图文混排外，还可以对页面进行整体布局，安排页面各个逻辑区域的位置。栏式结构是网页布局排版的基础结构，即采用竖分方式将页面上不同的逻辑区域分成若干列。例如 Theologisches Seminar Elstal 网站就采用了左右两栏的分栏式布局(见图 15-15)。

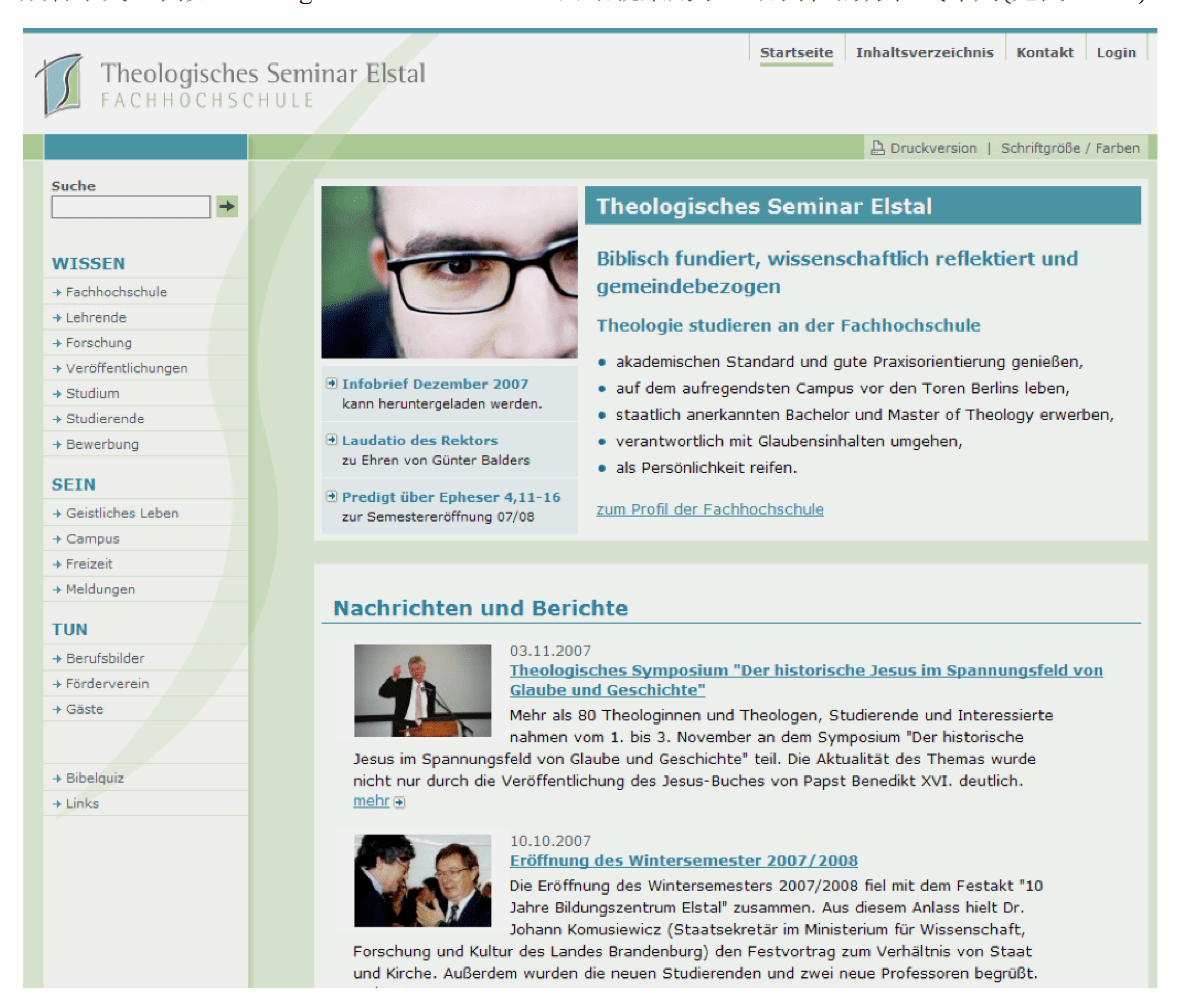


图 15-15 Theologisches Seminar Elstal 网站采用了左右分栏的布局

又如 Control C 网站采用了左中右三栏式的布局(见图 15-16)。

一般实现页面分栏式布局需要以下 3 个步骤来完成。

(1) 划分逻辑区域

首先将页面不同区域的内容放置在单独的 div 元素中，并设定好 id 属性(见图 15-17 的第一行左侧第一幅示意图)。

(2) 使侧栏浮动

根据需要可给侧栏添加 float 属性，使其向左或向右浮动。注意一点，在(X)HTML 文档中



浮动元素必须出现在围绕它的元素之前。最后给侧栏设定一定的宽度值。

图 15-16　Control C 网站采用了三栏式的页面布局

(3)　处理其余栏

对于其他栏，可继续添加 float 属性使其浮动，或者设定一定的边距，把覆盖在浮动元素之下的部分显示出来。

假如页面采用二分栏，左栏向左浮动，宽度为 200px。则第二栏会占据左栏原来的位置，且有一部分盖在左栏的下面，这时可设定它的 left-margin 属性为 210px，这样第二栏和左边这一栏就会相隔 10px。或者让第二栏也向左浮动并设置其左边距为 10px，效果是一样的。

图 15-17 展示了使用 float 属性进行分栏的原理。

本书的第 16 章将向读者介绍若干常见的页面布局形式和实现方式，那时我们再通过具体的代码来展示布局技术。



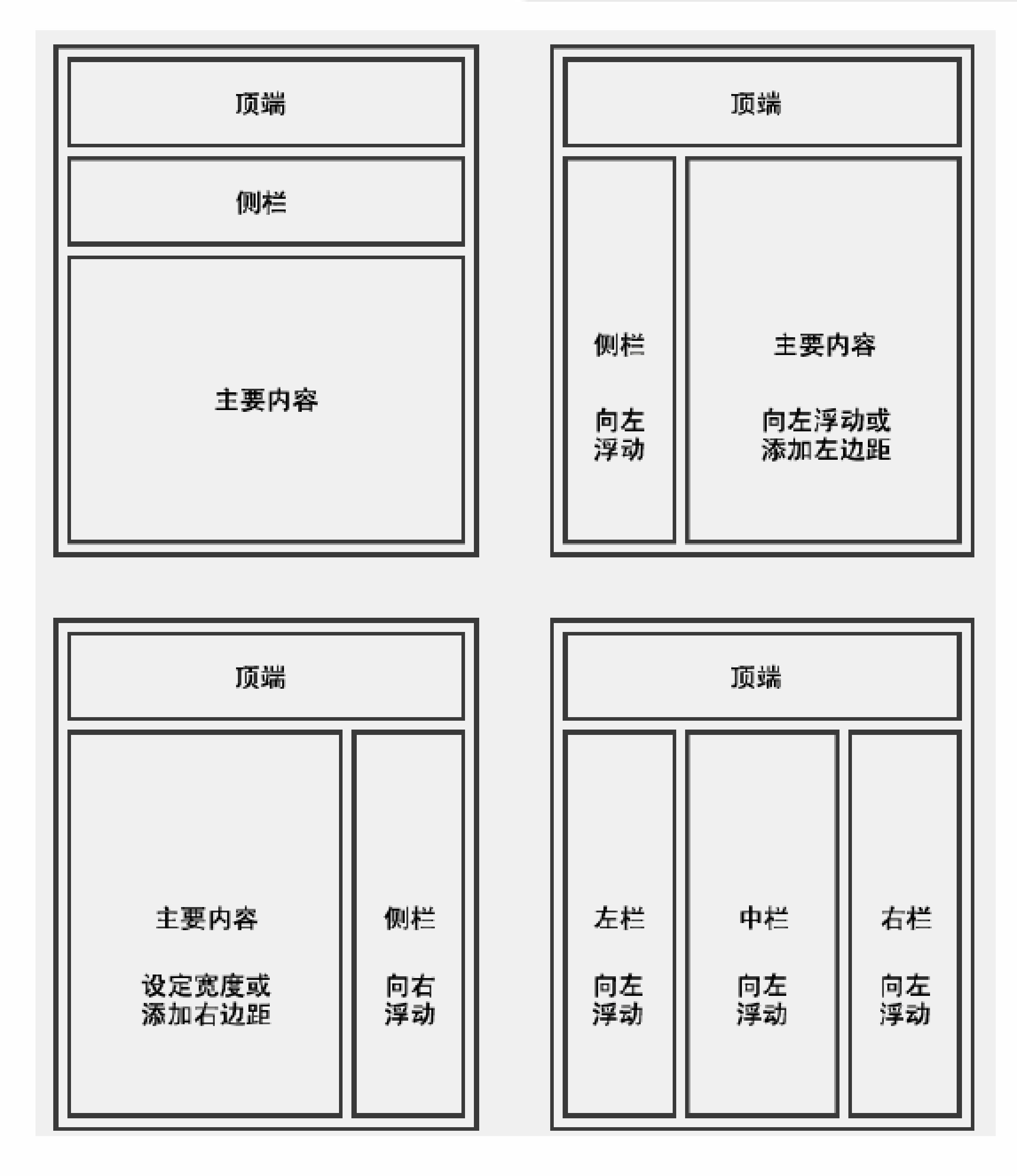


图 15-17　使用 float 属性实现页面分栏的原理示意图

15.4　IE 的浮动问题

Windows 平台下的 IE 浏览器存在许许多多的问题，与浮动相关的问题也不少，它直接影响页面布局的效果，如果不加以解决，会导致页面效果混乱不堪。

15.4.1　边距加倍问题

1. 问题描述及现象

IE6 以及之前的版本存在边距加倍问题(Double-margin Bug)。该问题的表现是，当元素向左或向右浮动时，元素的左边距或右边距的大小会被加倍。请看下面的示例：

```
body{
    margin:0;
}
div{
    background:yellow;
    border:1px solid gray;
    height:100px;
    width:100px;
    margin-top:10px;
}
```



```
div#box1{
    margin-left:50px;
    float:left;
}
div#box2{
    margin-right:50px;
    float:right;
}

<div id="box1"></div>
<div id="box2"></div>
```

我们准备两个 div 元素，使它们分别向左和向右浮动，并设置 50px 的左边距和右边距。通过对比 Firefox(图 15-18 上)和 IE6(图 15-18 下)的显示效果可以发现，IE6 中元素的边距被加倍了。

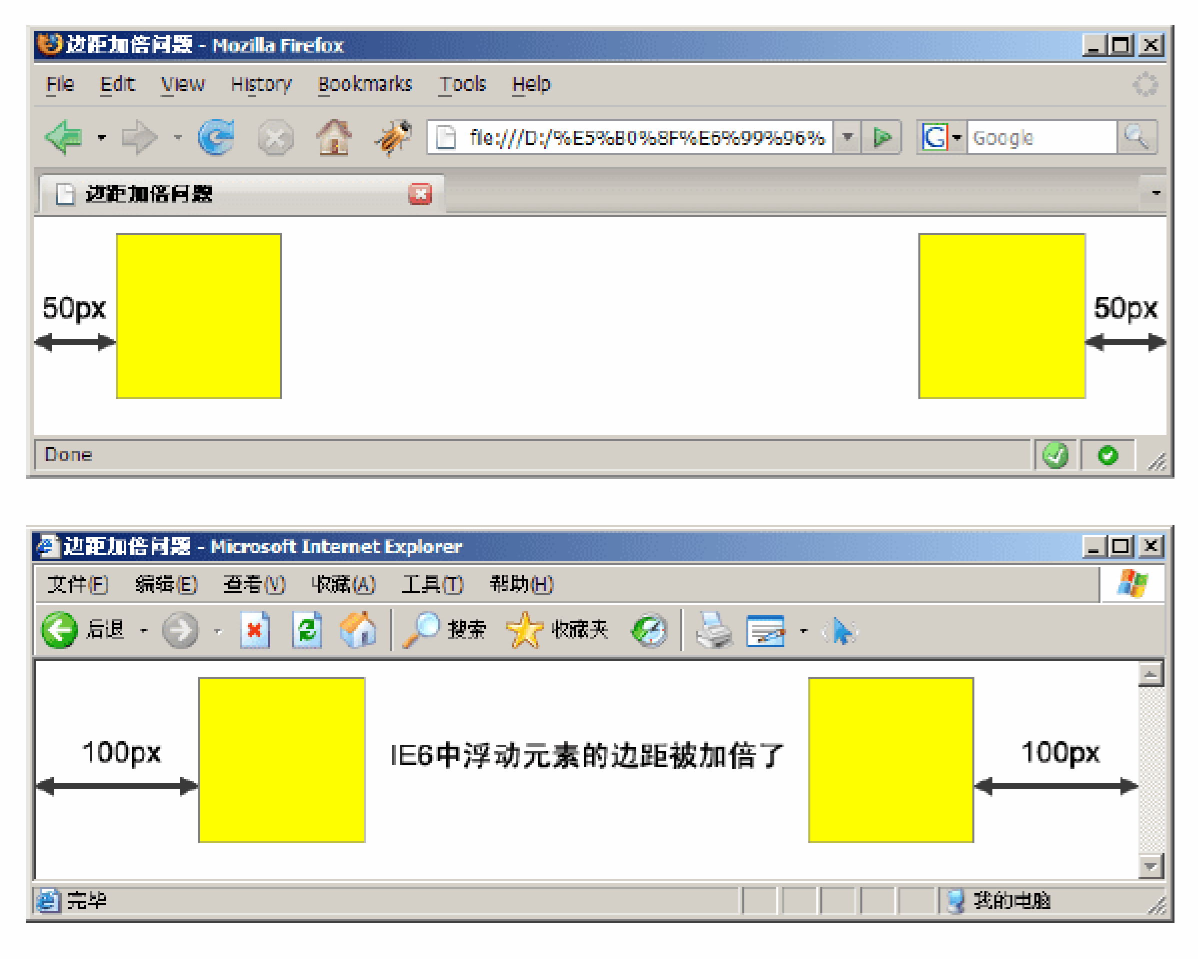


图 15-18　IE6 的边距加倍问题

2. 问题解决方案

该问题解决起来也很简单——只需要给浮动元素添加声明 display:inline 即可：

```
div#box1, div#box2{
    display:inline;
}
```

15.4.2　3px 间隔问题

1. 问题描述及现象

浮动元素会影响位于文档流之中元素的内容，使它们围绕在自身周围。但是在 IE6 中，浮动元素与周围的内容之间会自动多产生 3px 的距离，故称作 3px 间隔(Three-pixel Gap)问题或 3px 位移(Three-pixel Jog)问题。



请看示例：

```
div#box1{
    float:left;
    height:100px;
    width:100px;
    background:yellow;
    border:1px solid gray;
}
div#box2{
    margin-left:120px;
    border:1px solid gray;
}

<div id="box1"></div>
<p>测试文本</p>
<p>测试文本</p>
<div id="box2">
    测试文本<br />
    测试文本<br />
    测试文本<br />
    测试文本
</div>
```

代码创建了一个向左浮动的#box1，接着添加了两个 p 元素和一个 div 元素，元素中包含一些文本。我们给 div 元素设置了左边距，可以让读者更清楚地看到问题所在。图 15-19 展示了 IE6(上)和 Firefox(下)浏览器各自的显示效果。

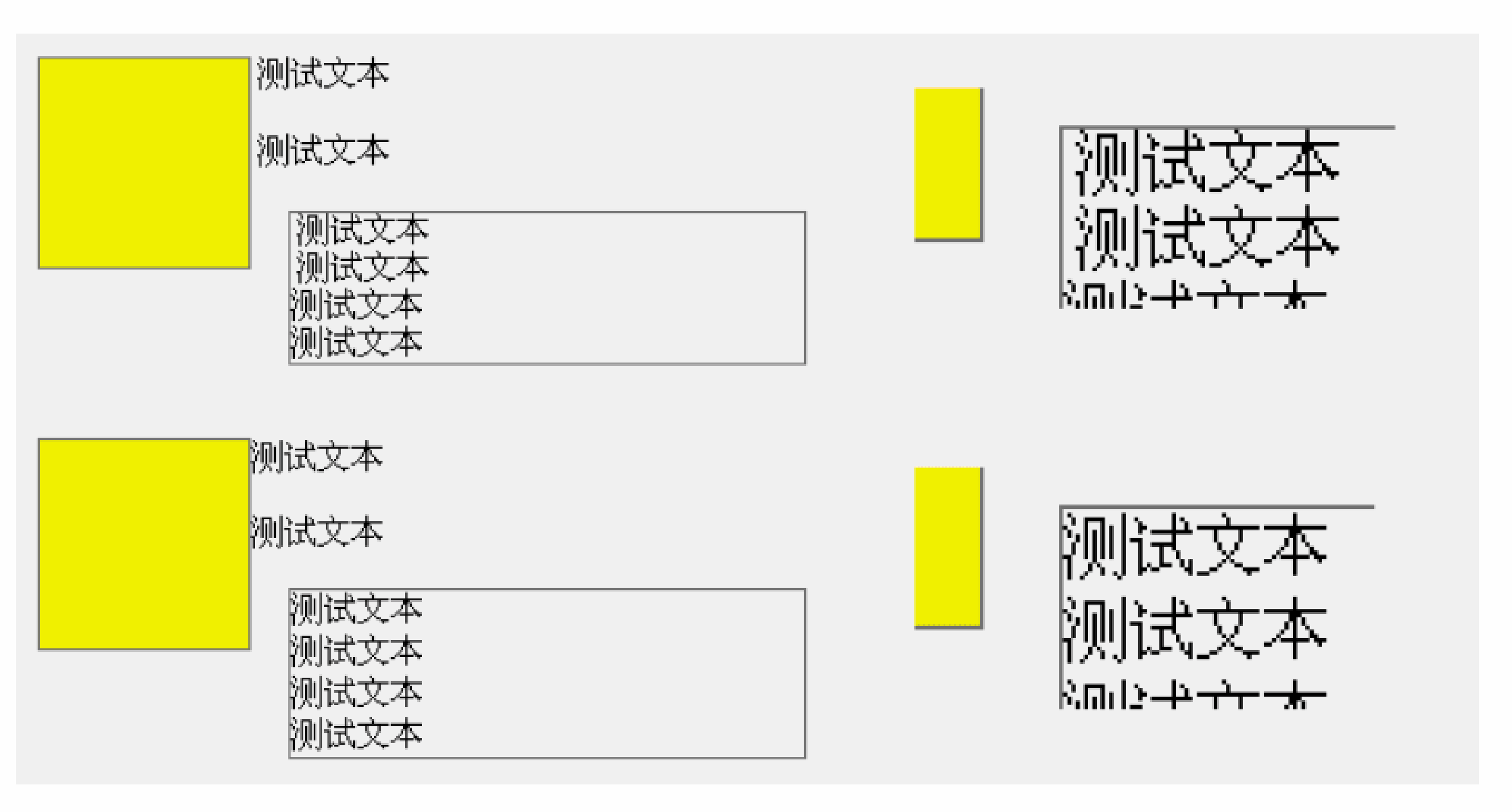


图 15-19　3px 间隔问题，IE6(上)中部分文本向右移动了 3px，Firefox(下)中没有此问题

可以看出，在浮动 div 的右侧，p 元素以及 div 元素中的文本与该浮动 div 之间存在一定的间隔，这个间隔恰好是 3px。而浮动 div 右下方的两行文本则没有这 3px 间隔。相当于浮动元素将右侧的元素挤出了 3px 的空间。



2. 问题解决方案

针对 3px 问题，我们可通过条件注释添加 IE6 专用的样式，将浮动元素右侧的边距减少 3px，可以使 p 元素文字向左移动 3px，给 div 元素添加 height:1px，使其内部文本左端对齐，但是这样做会使整个 div 元素中的内容全部增加 3px 的位移，我们再将它的左边距减少 3px，这样页面效果就和其余浏览器一致了。比如给#box1 添加如下声明：

```
<!-- 针对 IE6 的 3px 间隔问题进行修正 -->
<!--[if IE 6]>
<style type="text/css">
div#box1{
    margin-right:-3px;  /* 减少浮动元素的边距，0 - 3 = -3 */
}
div#box2{
    height:1px;         /* 使 div 元素内文本左端对齐 */
   margin-left:117px;   /* 向左移动 div 元素，120 - 3 = 117 */
}
</style>
<![endif]-->
```

提示： 对于一般的图文混排，3px 不会对页面效果产生很大的影响，是可以接受的。这时没有必要添加额外代码对 IE6 进行单独处理。假如浮动用于页面布局，且精度要求达到了像素级别，那么再考虑解决 3px 间隔问题。

15.4.3　捉迷藏问题

1. 问题描述及现象

捉迷藏问题(Peek-a-boo Bug)在很多情况下都会发生，尤其是当利用浮动进行页面布局时，div 元素中内容会不可见。请看示例：

```
div#wrapper{
    background:#DDD;
}
div#sidebar{
    float:left;
    border:1px solid gray;
    width:100px;
    height:150px;
    padding:10px;
}
div#main{
    border:1px solid gray;
}
div#hack{
    clear:both;
}
```

```
<div id="wrapper">
    <div id="sidebar">
        文本 <a href="#">链接</a>
    </div>
    <div id="main">
        文本<br />
        文本<a href="#">链接</a><br />
        文本<a href="#">链接</a>
        <div>文本<a href="#">链接</a></div>
    </div>
    <div id="hack">清除</div>
</div>
```

在 IE6 中打开该页面可以发现，#main 竟然消失了。当用鼠标选择该区域时，才能显示出其中内容，如图 15-20 所示。

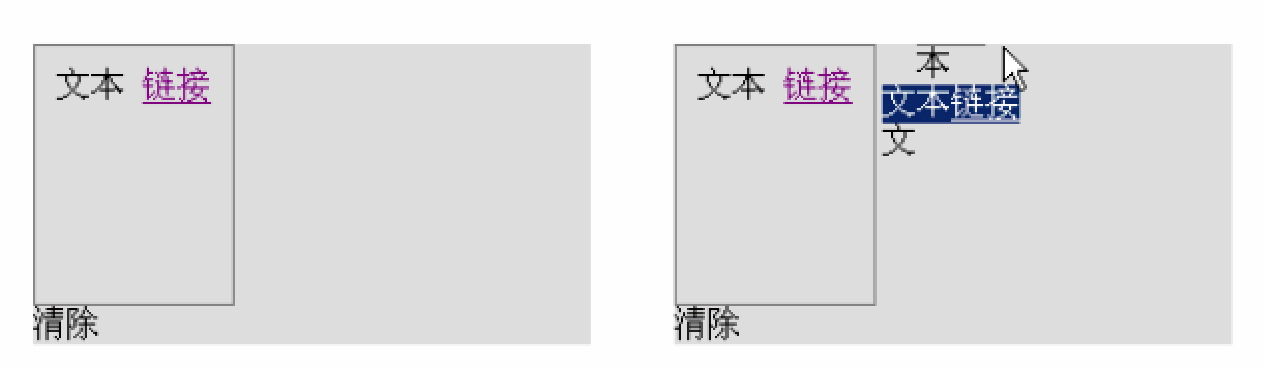


图 15-20　捉迷藏问题，IE6 没有显示右侧的元素(左图)，当用鼠标选择该区域时，元素即刻显示出来(右图)

捉迷藏问题产生的原因包含以下几个方面：

- 有浮动的元素存在。
- 容器有背景色。
- 有清除浮动影响的元素存在。
- 浮动元素周围的元素没有设置确切的高度和宽度值。

上述所有因素导致了 IE6 不能正确显示元素内容。

2. 问题解决方案

捉迷藏问题的解决办法也有很多：

- 给容器元素和浮动元素同时添加声明——position: relative;
- 避免给容器元素添加背景色。
- 给容器元素添加声明——zoom:1;
- 给容器元素添加声明——display:inline-block;
- 给容器元素或无法显示的元素指定宽度或高度值。

15.4.4　断头台问题

1. 问题描述及现象

断头台问题(Guillotine Bug)的名字起的很形象，当问题发生时，元素就如同上了断头台一样，其中的一部分内容会被砍掉。



请看示例：

```
div#wrapper{
    background:pink;
    border:1px solid black;
}
div#content{
    float:left;
    width:100px;
    border:1px solid red;
    margin:10px;
    background:gold;
}
ul{
    margin:0;
    padding:0;
    list-style:none;
}
a:hover{
    background:white;
}

<div id="wrapper">
    <div id="content">
        <p>文本 <a href="#">显示</a> 文本 文本 文本 文本</p>
        <p>文本 文本 文本 文本 文本 文本 文本</p>
    </div>
    <ul>
        <li><a href="#">显示</a></li>
        <li><a href="#">显示</a></li>
        <li><a href="#">不显示</a></li>
        <li><a href="#">不显示</a></li>
        <li><a href="#">不显示</a></li>
    </ul>
</div>
```

代码包含一个容器#wrapper，里面有个浮动的 div 和若干链接。当用 IE6 打开这个页面时，页面显示正常，当鼠标移至那些“不显示”的链接时，问题出现了，浮动元素的下部被砍掉不见了，当鼠标移至“显示”的链接时，浮动元素的高度又恢复正常，如图 15-21 所示。

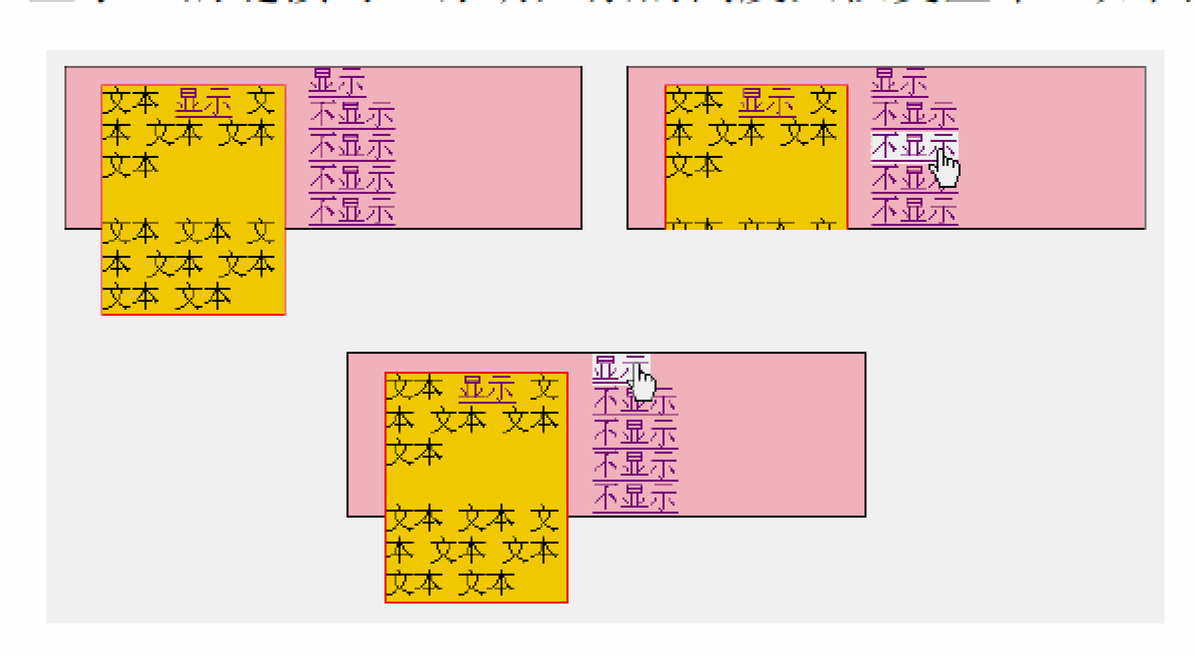


图 15-21　断头台问题



导致断头台问题出现的原因包含以下几方面：

- 有浮动元素存在于容器中。
- 容器中存在非浮动的链接。
- 通过:hover 伪类改变链接的某些属性。

2. 问题解决方案

可以根据断头台问题产生的原因进行解决，把非浮动对象改为浮动对象，或者添加额外元素清除浮动(注意不要引入捉迷藏问题)。

作为最常使用的浏览器，IE6 存在的问题也很多。读者在实际开发中，每完成一部分样式编码后就该用浏览器验证页面效果，以便能及早发现并解决问题。

延伸： 在实际编写(X)HTML 和 CSS 样式代码的过程中，每完成一小部分就应该立刻用浏览器打开页面查看效果，这在敏捷开发中称为小步迭代。这样做使我们能够在短时间内发现问题，且问题范围较小，解决起来比较容易。

15.5 position 属性与定位

position 属性可用来控制元素如何显示以及在什么位置显示，它的属性值可以为 static、relative、absolute 和 fixed，其中默认值为 static，拥有非默认值的元素称为已定位元素(Positioned Element)或非静态定位元素。position 各属性值分别代表着 CSS 提供的如下 4 种类型的定位。

- 静态定位(static)：元素将按照文档流的默认顺序显示。
- 相对定位(relative)：元素可以在文档流中的原始位置上增加偏移量，即相对于文档流中的位置进行定位。
- 绝对定位(absolute)：元素将脱离文档流，通过上下左右偏移值的组合来确定该元素的位置。
- 固定定位(fixed)：元素将固定在某一确定的位置上，即使页面滚动也不会影响它的位置。

现在就来依次了解各种定位类型的含义和用法。

15.5.1 静态定位和相对定位

静态定位是(X)HTML 文档中各个元素默认的定位方式，它们将按照文档流的默认方式显示。在正常的文档流中，每个元素都被定位到某个位置，我们称其为“起始点”，此时 top、right、bottom 和 left 属性都不起作用。而相对定位则是在这个起始点的基础上通过 top、right、bottom 和 left 属性的组合对元素进行移位。请看下面的示例：

```
body{
    padding:10px;
    border:1px dashed #999;
}
```



```
div{
    height:80px;
    width:100px;
    border:1px solid gray;
    background:yellow;
}
.relative{
    position:relative;
}
div#box2{
    left:10px;
    top:10px;
}
div#box3{
    right:10px;
    bottom:10px;
}

<div class="static" id="box1">box1 静态定位</div>
<div class="relative" id="box2">box2 相对定位</div>
<div class="relative" id="box3">box3 相对定位</div>
```

如图 15-22 所示为以上代码的效果。



图 15-22　静态定位与相对定位

#box2 和#box3 采用相对定位，通过设定 left 和 top 属性让#box2 向右向下各偏移 10px，设定 right 和 bottom 属性让#box3 向左向上各偏移 10px。相对定位中的偏移与静态定位中的 margin 属性的作用类似。

图 15-23 为相对定位的原理示意图。

需要注意的是，偏移值的组合只能是以下 4 种之一，其余的组合都无效。

这 4 种组合为：①top 和 left；②top 和 right；③bottom 和 left；④bottom 和 right。即水平和垂直方向各取一个属性值进行组合。如果水平或垂直方向存在两个相互排斥的属性，不管它们的顺序如何，则只有 left 和 top 属性起作用。



例如：

```
div{
    left:10px;
    right:10px;
    top:10px;
    bottom:10px;
}
```

相当于：

```
div{
    left:10px;
    top:10px;
}
```

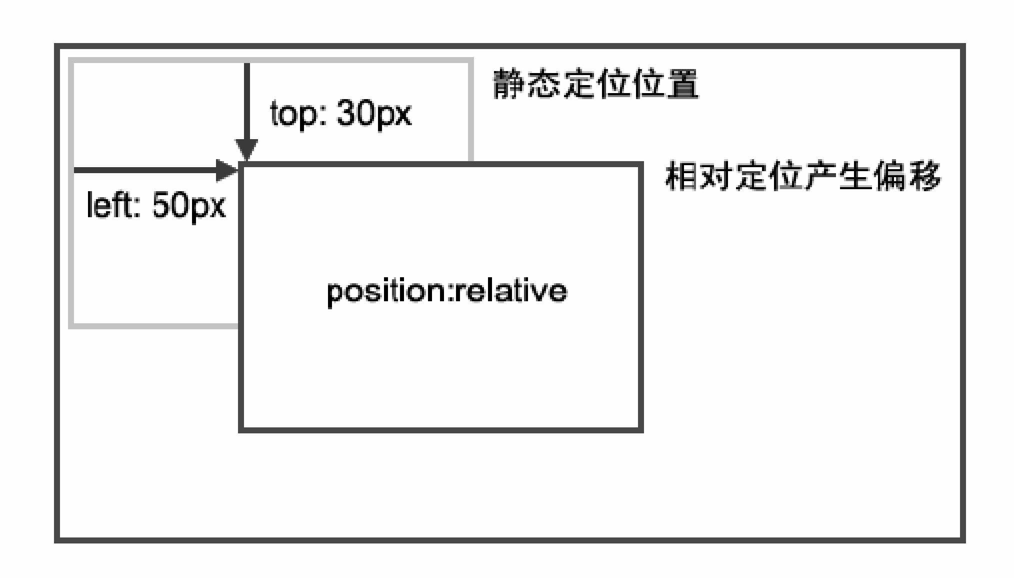


图 15-23　相对定位的原理示意图

15.5.2　绝对定位

绝对定位的元素将脱离原始文档流，这一点和浮动元素的特点相同，但是绝对定位的元素对其他元素不会产生任何影响，其他元素对其“视而不见”，仿佛它根本不存在。通过 top、right、bottom 和 left 四个方向的偏移组合可控制绝对定位元素的位置。

我们通过几个示例来说明：

```
*{
    padding:0;
    margin:0;
}
div{
    height:80px;
    width:100px;
    border:1px solid gray;
    background:yellow;
}
.absolute{
    position:absolute;
}
```



```
div#box1{
    left:50px;
    top:30px;
}
div#box3{
    left:0;
    bottom:0;
}
div#box4{
    right:0;
    top:0;
}
div#box2{
    margin-bottom:20px;
}
p{
    text-indent:2em;
    line-height:1.8em;
    width:400px;
}

<div id="box1" class="absolute">box1 绝对定位</div>
<div id="box2">box2 静态定位</div>
<div id="box3" class="absolute">box3 绝对定位</div>
<div id="box4" class="absolute">box4 绝对定位</div>
<p>绝对定位的元素将脱离文档流，这一点和浮动元素的特点相同，但是绝对定位的元素不会对其他元素产生任何影响，文档流中的元素对其“视而不见”，仿佛它并不存在。</p>
<p>通过设置 top、right、bottom 和 left 属性值可以精确控制绝对定位元素的偏移位置。</p>
```

以上代码的效果如图 15-24 所示。

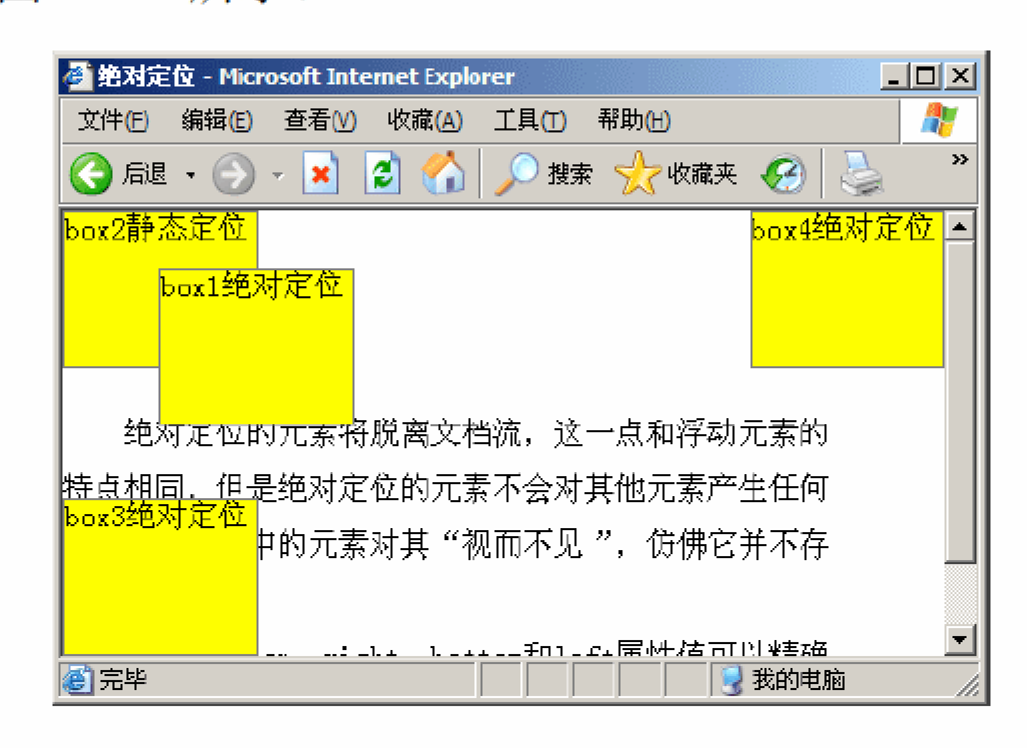


图 15-24　绝对定位

从图 15-24 中可以看出，由于#box1、#box3 和#box4 采用了绝对定位，它们脱离了文档流，而余下的#box2 和两个 p 元素则按照默认方式显示。绝对定位的元素会覆盖在其余元素的上面(下一小节介绍的深度属性可以将其置于元素下方，我们后面再做具体的介绍)，但是对元素的



内容没有产生任何影响。#box1 设置了 left 和 top 属性，则它相对于页面的可视区域(Viewport)左上方产生偏移。同理，#box3 设置了 left 和 bottom，它相对于可视区域的左下方产生偏移。#box4 则相对于可视区域的右上方产生偏移。也就是说，这时的绝对定位元素是相对于可视区域进行移位。当我们滚动浏览器的滚动条时，绝对定位的元素会跟随可视区域中的文档一起移动，如图 15-25 所示。

提示： 可视区域(Viewport)，也称作视见区或视口，它是浏览器为用户提供的一个用于查看 Web 文档的区域，通常就是浏览器显示网页的矩形区域。如果文档内容超过了这个区域的大小，浏览器会提供一种滚动机制。比如当网页内容超过窗口当前大小时，浏览器会自动出现横向或/和纵向的滚动条。

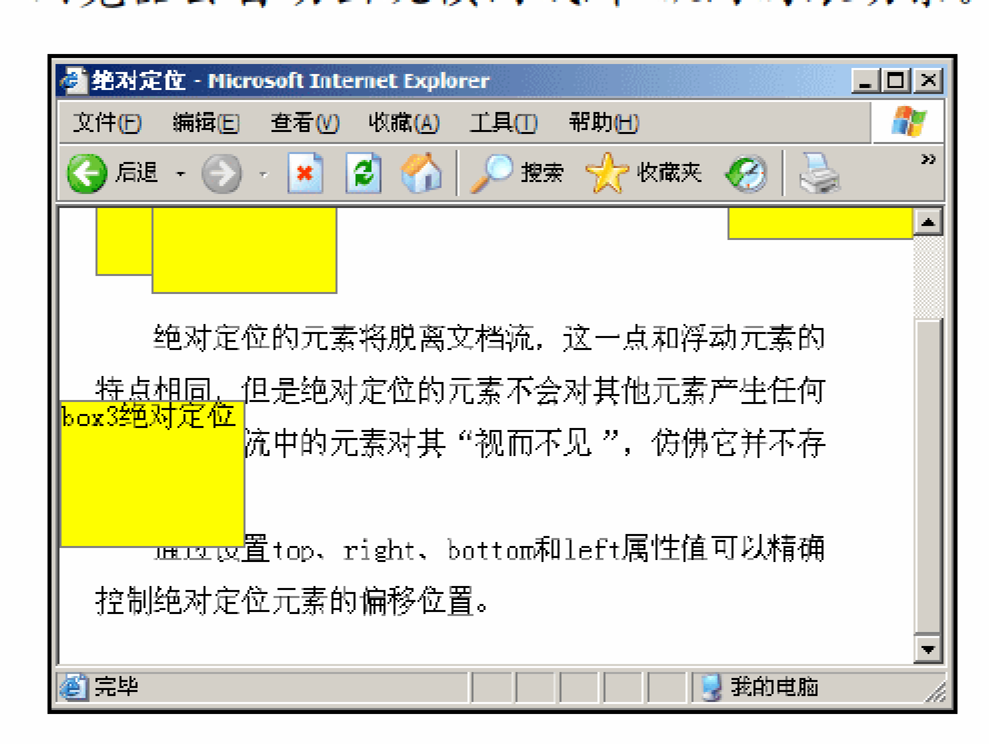


图 15-25　绝对定位元素会跟随可视区域中的文档一起移动

一般情况下，我们会给绝对定位的元素设置确切的宽度和高度值，然后从水平和垂直方向中各取一个属性来为元素提供位置信息，这和控制相对定位元素的位置是一样的。如果位置信息存在冲突，那么只有 left 和 top 属性起作用。

请看示例：

```
.absolute{
    position:absolute;
}
div#box1{
    padding:10px;
    border:1px solid black;
    background:#AAA;
    width:150px;
    height:100px;
    top:10px;
    bottom:10px;
    right:10px;
    left:10px;
}

<div id="box1" class="absolute">绝对定位</div>
```



以上代码给#box1 设置了 4 个方向的位移值，但是只有 left 和 top 起作用，因此#box1 将距离可视区域的左端和顶端各 10px(见图 15-26)。

但是，当绝对定位元素没有被指定确切的高度或宽度时，偏移属性将直接控制元素边界的位置，从而能起到控制元素的大小的作用。

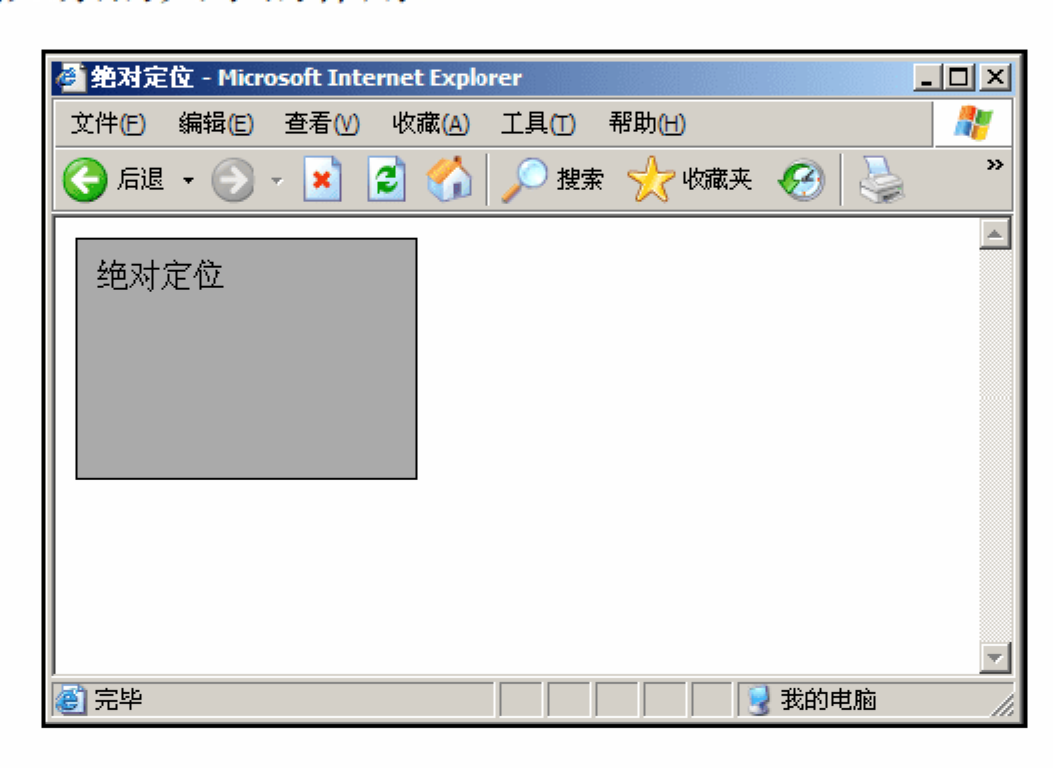


图 15-26　当偏移属性产生冲突时，绝对定位元素的位置由 left 和 top 属性确定

图 15-27 中的#box1 四个方向的边界距离可视区域的四边均是 10px，同时也改变了#box1 的大小。

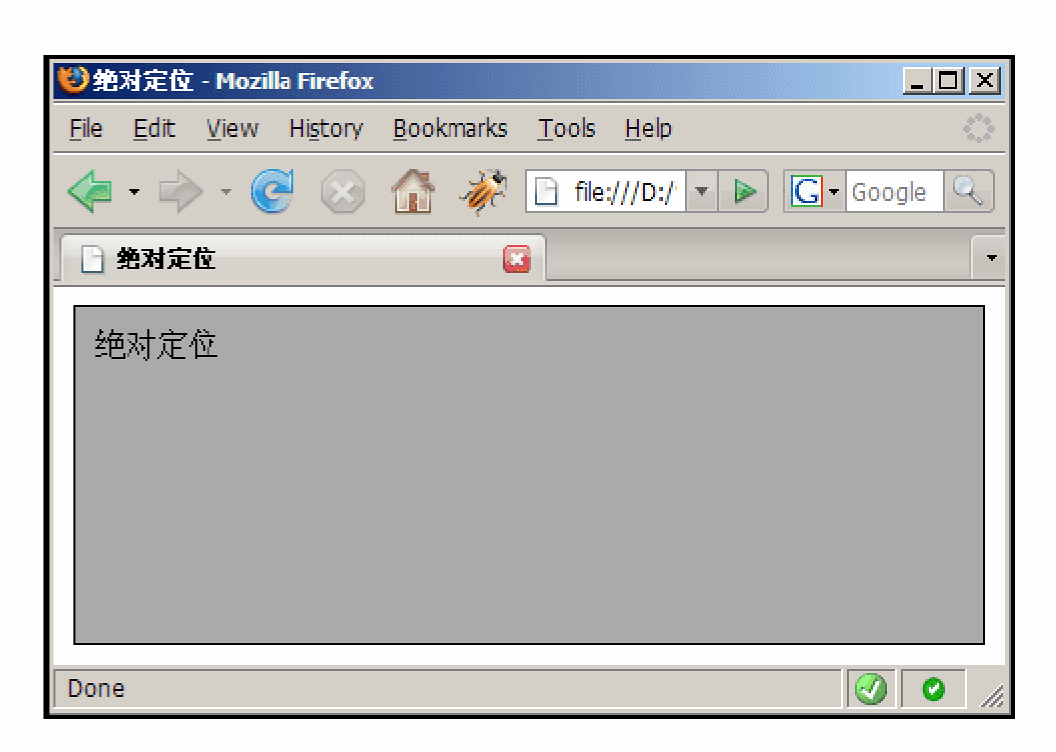
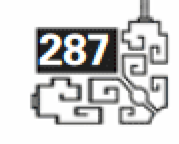


图 15-27　通过偏移属性控制元素大小

到目前为止，我们看到绝对定位元素都是在浏览器窗口中确定位置的，除此之外，绝对定位元素还可以相对于其某个祖先元素进行定位，该祖先元素是距离这个元素最近的已定位元素。请看下面这个示例：

```
div#container{
    width:200px;
    height:160px;
    border:1px solid gray;
}
.absolute{
    position:absolute;
}
```



```
div#box1{
    height:80px;
    width:120px;
    left:0;
    top:0;
    border:1px solid black;
    background:yellow;
}

<div id="container">
    <div id="box1" class="absolute">绝对定位</div>
</div>
```

以上代码效果如图 15-28 所示，#box1 脱离了父元素#container 的包围，紧靠在浏览器窗口的左上角。

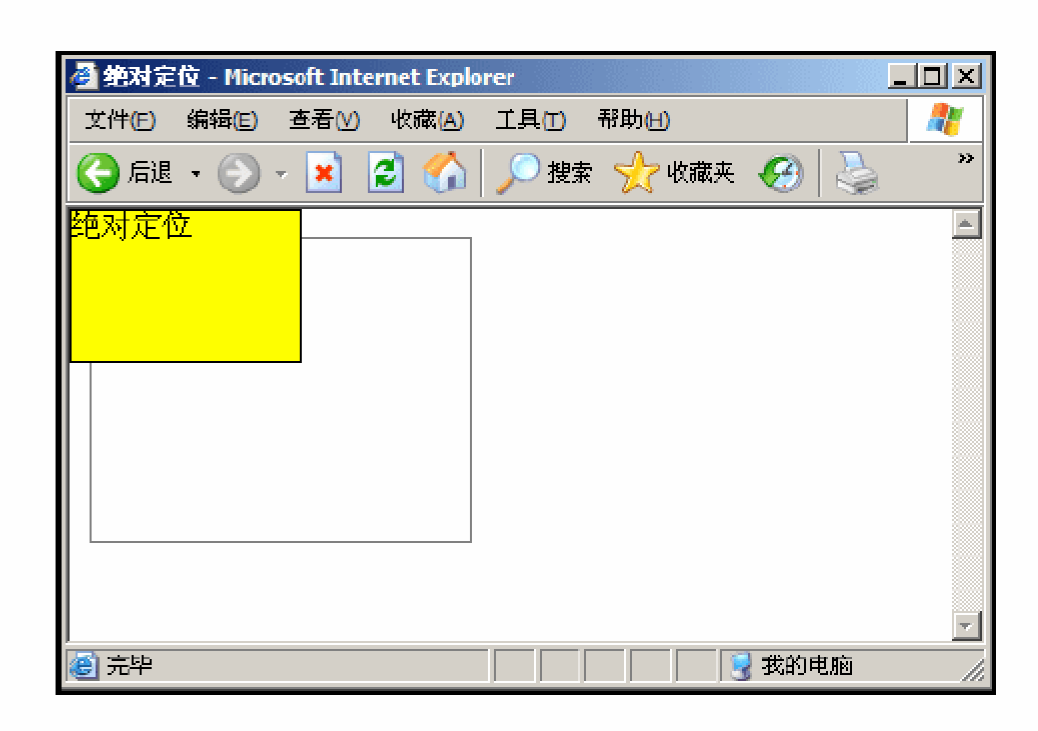


图 15-28　绝对定位元素脱离了父元素的包围

如果将#container 的 position 属性设为 relative、absolute 或 fixed(IE6 中无效)，则#box1 将相对于#container 进行定位(见图 15-29)。

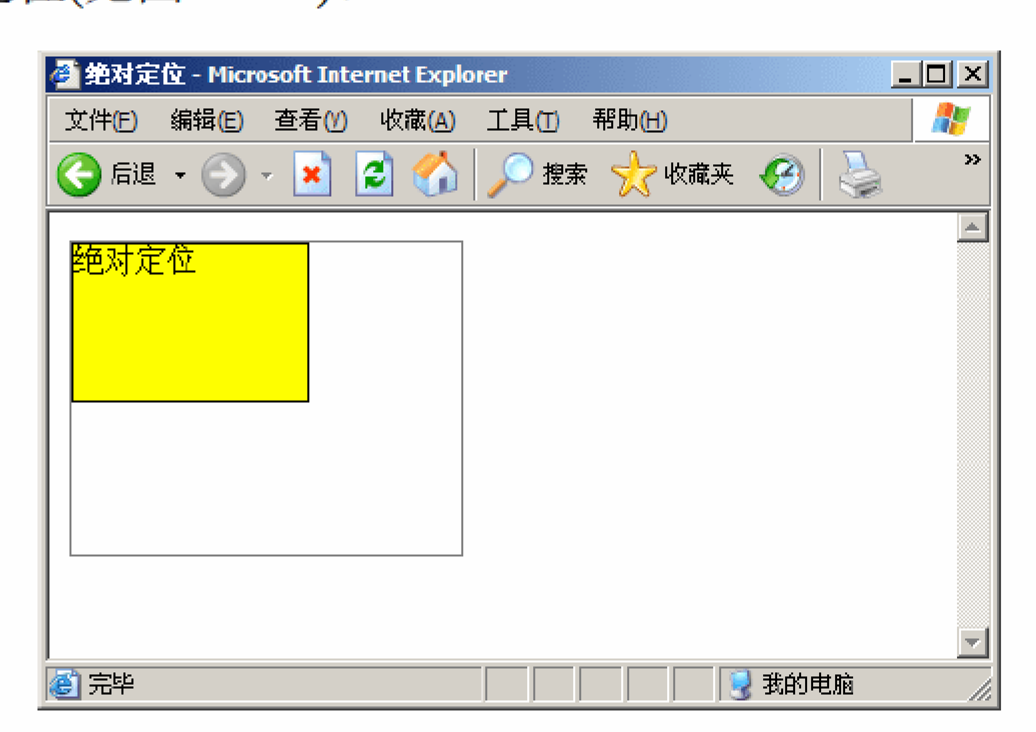


图 15-29　绝对定位元素相对于距离自己最近的已定位的父元素进行定位

图 15-30 为绝对定位的原理示意图。



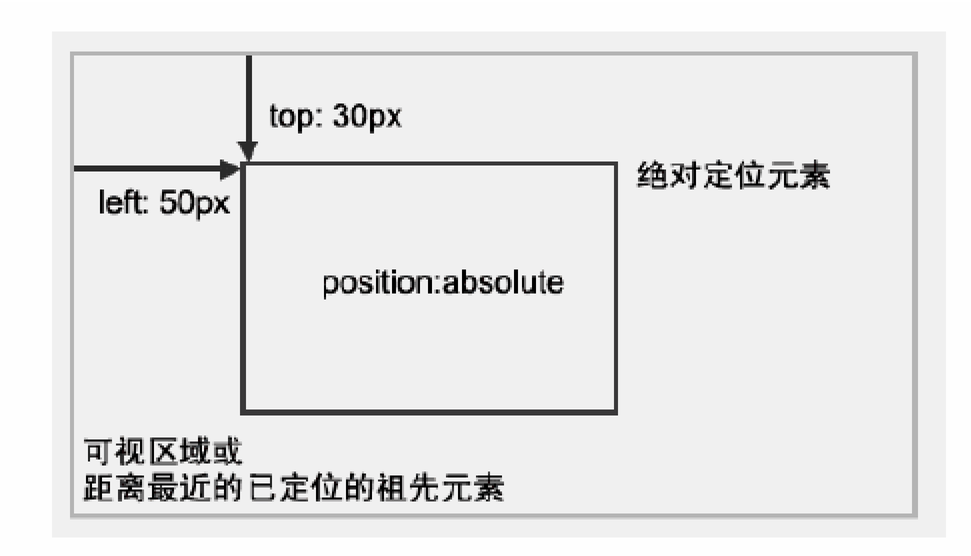


图 15-30　绝对定位的原理示意图

15.5.3　固定定位

固定定位的元素会被固定在浏览器中，不会跟随页面一起滚动。注意 IE6 不支持固定定位。请看下面这个示例：

```
p{
    text-indent:2em;
    line-height:2em;
}
.fixed{
    position:fixed;
}
div#box1{
    width:100px;
    height:100px;
    border:1px solid gray;
    background:silver;
}

<div id="box1" class="fixed"></div>
<p>固定定位元素将被固定浏览器窗口中，不会跟随页面滚动而发生位移。这和
background-attachment 属性的 fixed 效果类似。IE6 及以下版本不支持固定定位。</p>
<p>固定定位元素将被固定浏览器窗口中，不会跟随页面滚动而发生位移。这和
background-attachment 属性的 fixed 效果类似。IE6 及以下版本不支持固定定位。</p>
<p>固定定位元素将被固定浏览器窗口中，不会跟随页面滚动而发生位移。这和
background-attachment 属性的 fixed 效果类似。IE6 及以下版本不支持固定定位。</p>
```

代码效果如图 15-31 所示。

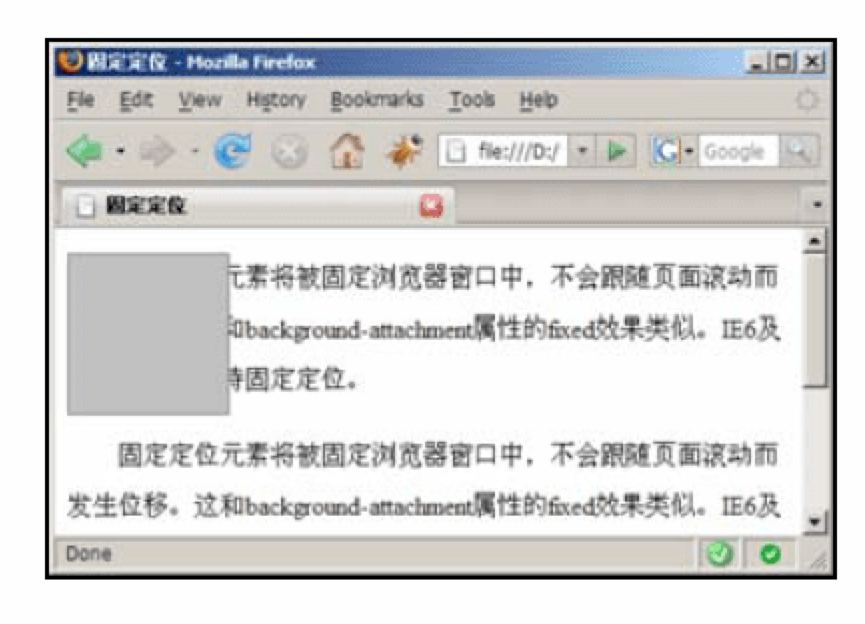
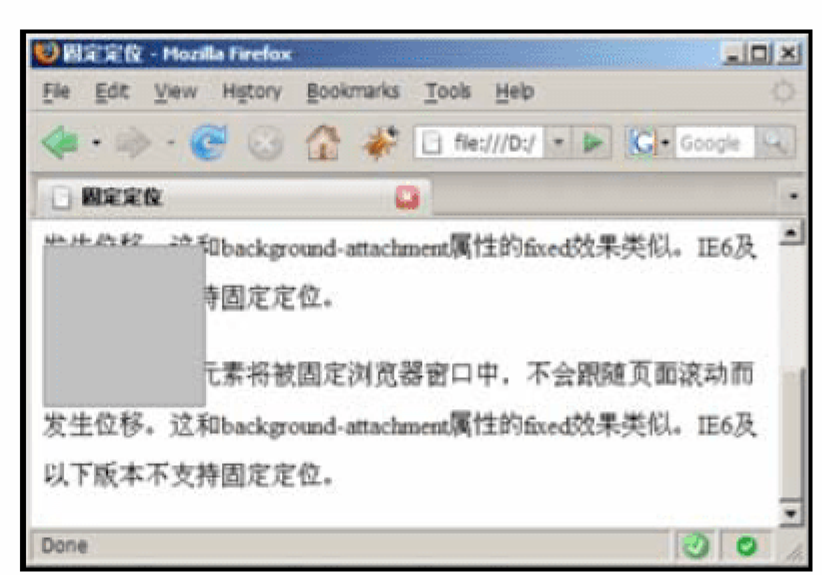
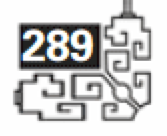



图 15-31　固定定位元素，其位置永远基于浏览器的窗口(Firefox 浏览器)



固定定位元素的 left、right、top 和 bottom 属性的用法和意义与绝对定位元素一致，这里不再赘述。

15.6 利用定位实现布局

和浮动属性相比，利用位置属性进行页面布局的情况在实际中使用得并不多见，但是它的确可以进行页面布局，本节我们将介绍如何利用 position 属性实现分栏式的布局。

利用定位实现分栏式布局需要如下几个步骤。

(1) 划分逻辑区域

首先将页面不同区域的内容放置在单独的 div 元素中，并设定好 ID 属性。有时可能还会增加额外的 div，作为其他元素的容器。

(2) 设置侧栏的位置属性

将作为容器使用的 div 元素的 position 属性设为 relative，目的是让其内部元素能正确定位。然后将侧栏的 position 属性设为 absolute，并根据需要确定其位置。

(3) 处理其余栏

对于其余栏，可根据需要设置左右边距，把覆盖在侧栏下面的内容显示出来。

上述布局步骤可参考图 15-32。





图 15-32　利用 position 属性实现分栏式布局的原理图

下面这段代码展示了一个三分栏式的布局结构。

```
div{
    border:1px solid gray;
    background:yellow;
}
#top{
    width:700px;
    height:40px;
    margin-bottom:10px;
}
```



```
#container{
    position:relative;
    width:702px;
    background:none;
    border:none;
}
#sidebar1, #sidebar2{
    width:180px;
    position:absolute;
    top:0;
    background:pink;
    height:280px;
}
#sidebar1{
    left:0;
}
#sidebar2{
    right:0;
}
#content{
    margin:0 190px;
    background:pink;
    height:320px;
}
#footer{
    margin-top:10px;
    width:700px;
    height:40px;
}

<div id="top">顶端</div>
<div id="container">
   <div id="sidebar1">左栏</div>
   <div id="content">中栏</div>
   <div id="sidebar2">右栏</div>
</div>
<div id="footer">底脚</div>
```

效果如图 15-33 所示。

注意： 该例中，如果左栏或右栏的高度超过了中栏，则左栏或右栏会超出容器元素覆盖到下面的底脚，这是因为绝对定位的元素脱离了文档流，它的容器不再将其包含在内，高度也就不会随之变化。这也是使用 position 属性定位所带来的问题。可以添加额外的脚本代码来控制容器元素的高度。



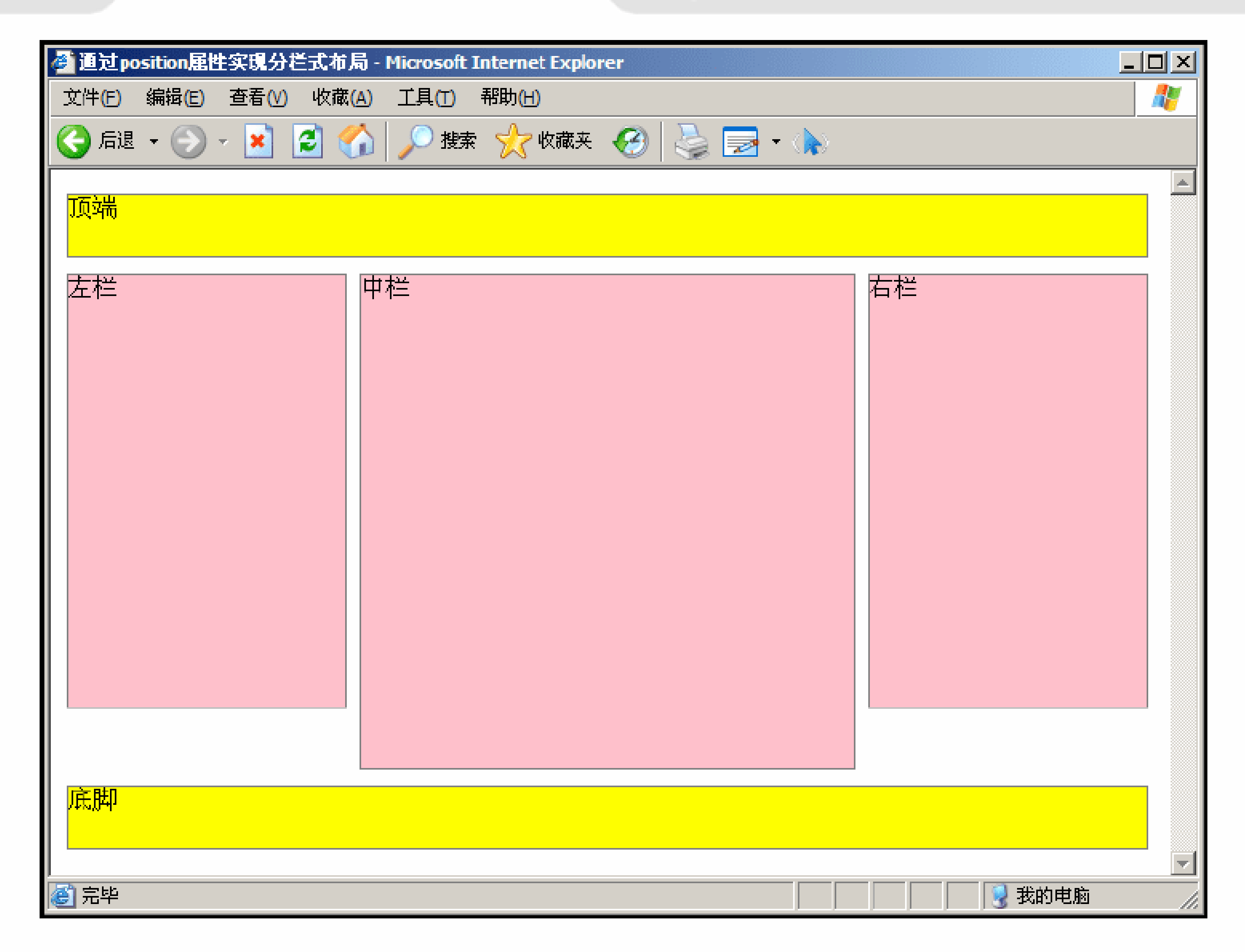


图 15-33　通过 position 属性实现三栏式布局

15.7　控制元素的深度

在前面的示例中，任何已定位元素都会覆盖在位于文档流中的元素之上，实际上每个元素分别位于深度不同的层中，当层与层之间有重叠区域时，就会有一个元素覆盖到另一个元素之上。相当于在平面的 x 轴和 y 轴基础上添加了一个虚拟的垂直于浏览器的 z 轴。CSS 的 z-index 属性可修改元素的深度属性，调整覆盖关系。z-index 属性值为整数，数值越大越靠近 z 轴的顶端(越靠近浏览者)。注意 z-index 属性只对已定位元素有效。

请看示例：

```
div{
    height:100px;
    width:100px;
    border:1px solid gray;
    position:absolute;
}
div#box1{
    left:10px;
    top:10px;
    background:yellow;
}
div#box2{
    left:20px;
    top:20px;
    background:green;
}
```



```
div#box3{
    left:30px;
    top:30px;
    background:pink;
}
div#box4{
    left:40px;
    top:40px;
    background:red;
}

<div id="box1"></div>
<div id="box2"></div>
<div id="box3"></div>
<div id="box4"></div>
```

默认情况下，后面的元素会覆盖在前面的元素之上(如图 15-34 所示)。

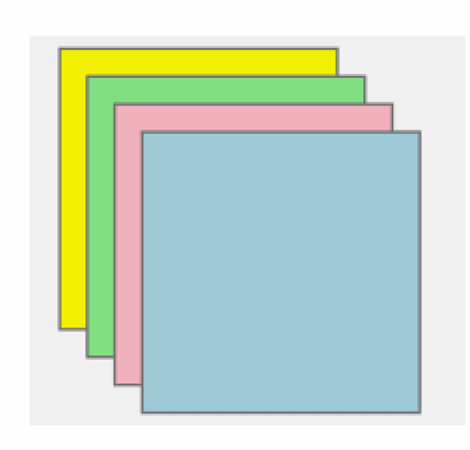


图 15-34　默认情况下后面的元素覆盖前一元素上

现在我们给 4 个 div 元素分别添加 z-index 属性：

```
div#box1{z-index:4;}
div#box2{z-index:3;}
div#box3{z-index:2;}
div#box4{z-index:1;}
```

#box1 位于最顶端，下面依次是#box2、#box3 和#box4(如图 15-35 所示)。

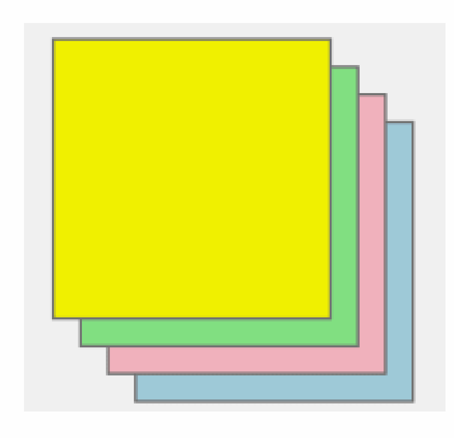


图 15-35　通过 z-index 属性修改元素深度

一些网页提供了类似于操作系统中的提示框的功能，该提示框会位于其余任何元素之上，这就是通过设定一个相当大的 z-index 属性值实现的。

提示： 在 IE 中，表单元素 select 不受 z-index 元素的影响，总是位于其他元素之上。可以在需要时将 select 暂时隐藏掉。



15.8 元素的可见性

CSS 的 visibility 属性可控制元素是否可见，属性值 visible 表示可见，hidden 则表示隐藏。它经常配合不同类型的已定位元素使用。比如上一小节提到的提示框，不显示时可将其设为 hidden，当需要向访问者提示信息时可将其设为 visible(通过脚本代码实现转换)。

visibility 的 hidden 和 display 的 none 类似，都是不显示元素，但它们的实现方式略有不同。hidden 将元素隐藏，但是元素所占据的空间依旧存在。none 相当于将元素完全除掉。

请看示例：

```
<p>visibility 的 hidden 与 display 的 none 类似，但是有所区别。</p>
<p>visibility 的<span style="visibility:hidden;">hidden</span>与 display 的
<span style="display:none;">none</span>类似，但是有所区别。</p>
```

如图 15-36 所示为以上代码的效果。

visibility的hidden与display的none类似，但是有所区别。

visibility的　　　与display的类似，但是有所区别。

图 15-36　hidden 和 none 的区别

第 7 章讲过，overflow 属性告诉浏览器如何处理超出元素范围的区域。但是绝对定位的元素会脱离文档流，使父元素的 overflow 属性不再起作用(前提是父元素采用默认的静态定位)。这时我们可以使用 CSS 的 clip 属性对绝对定位元素进行裁剪。

clip 属性的形式为：

```
clip: rect(top right bottom left);
```

top、right、bottom 和 left 分别表示上、右、下、左四个方向的边相对于元素左上角的偏移值，请看示例：

```
body{
    margin:10px;
}
div#box1{
    height:100px;
    width:100px;
    background:yellow;
    overflow:hidden;
}
div#box2{
    height:60px;
    width:200px;
    position:relative;
    top:10px;
    left:10px;
    background:pink;
```



```
}

<div id="box1">
    <div id="box2"></div>
</div>
```

#box2 采用相对定位，它超出#box1 的区域不可见，图 15-37 显示了 Firefox 浏览器中的效果，注意 IE 浏览器中 overflow 属性不再起作用。

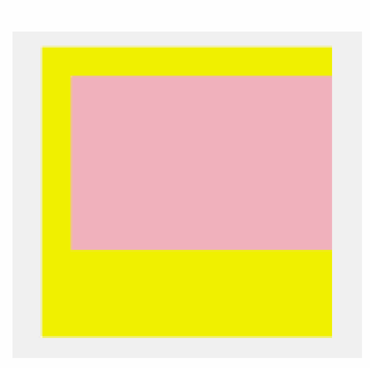


图 15-37　overflow 属性对采用相对定位的元素依然有效(Firefox 浏览器)

如果我们将#box2 的 position 属性改为 absolute，那么 overflow 属性将不起作用，除非改变#box1 默认的定位属性。图 15-38 展示了修改#box2 的 position 属性之后的效果。

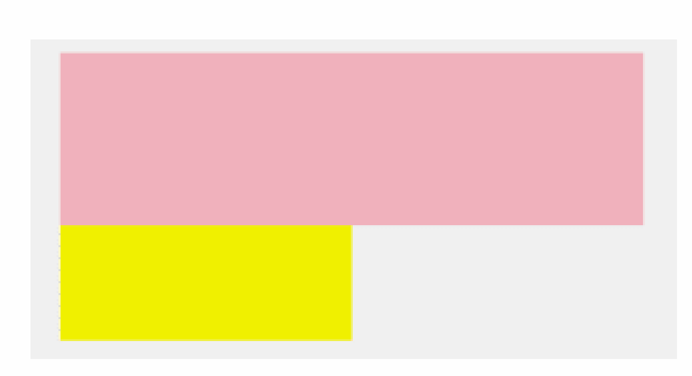


图 15-38　绝对定位元素使采用静态定位的父元素的 overflow 属性失效

现在我们为#box2 添加 clip 属性对其进行剪裁：

```
clip: rect(0 100px 60px 0);
```

#box2 的右边距离左上角的水平距离为 100px，这样就从视觉上隐藏了超出的部分(见图 15-39)。当然我们也可以任意调整裁剪区域的大小以实现其他效果。

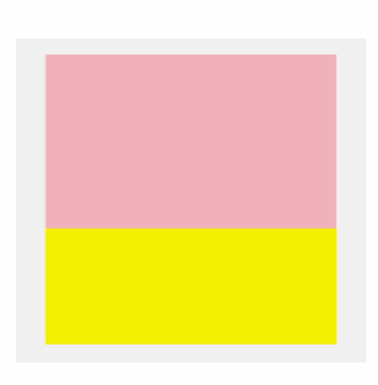


图 15-39　通过 clip 属性对#box2 进行裁剪

注意：如果读者查阅 CSS 规范或者其他一些介绍 CSS 的书籍，就会发现 clip 属性的形式为 “rect(top, right, bottom, left)”，各个裁剪范围值之间用逗号相隔，而在本书中，clip 属性的各个值之间用空格相隔。实践证明，IE 浏览器只能识别以空白(包括空格符、制表符和换行符)相隔的 clip 属性值，因此本书采用的形式为 “rect(top right bottom left)”。



15.9 小　结

本章介绍了 CSS 布局中常用的浮动技术。(X)HTML 中各种元素遵循着一定的显示规律：内联元素沿水平方向依次排列，元素前后没有换行；块级元素沿竖直方向依次排列，元素前后产生换行，这种显示方式就构成了(X)HTML 文档流。

浮动元素会脱离原始的(X)HTML 文档流，“漂浮”在这个文档流之上，后面的非浮动元素会占据浮动元素的位置，且非浮动元素中的内容会围绕在浮动元素周围。元素浮动后将自动转换为块级元素，它向左或向右停靠在其容器的边缘。

float 属性可以使元素产生向左或向右的浮动，使用它可以轻松实现图文混排。clear 属性可以消除浮动元素对非浮动元素产生的影响，即非浮动元素中的内容会围绕在浮动元素的周围，使用 clear 属性可以取消这种围绕效果。

浮动是页面布局的基础，利用 float 属性可以实现页面分栏式布局。

本章还介绍了 IE 浏览器中 4 个有关浮动的问题、它们产生的原因以及相应的解决方法。

CSS 的 position 属性可控制元素的定位方式，默认情况下(X)HTML 中元素采用静态定位方式，即按照文档流的顺序进行显示。除此之外，元素还可以进行相对定位、绝对定位和固定定位。

相对定位的元素以文档流中默认位置为基准进行位移。绝对定位元素将脱离原始文档流，它将相对于距离自己最近的已定位的祖先元素进行定位，如果不存在这样的祖先元素，则大多数浏览器会使其相对于可视区域进行定位。固定定位元素始终保持其在浏览器可视区域中的位置，不受页面滚动的影响。注意 IE6 不支持固定定位。

最后还介绍了如何利用 position 属性实现页面分栏式的布局，并通过具体的代码进行了展示。

CSS 的 z-index 属性可控制元素的深度，即 z 轴上的位置。visibility 属性可用来控制元素的可见性。clip 属性可对绝对定位的元素进行裁剪。

在下一章中，将介绍几种常见的页面布局形式以及它们的实现方法。

第 16 章　常见的页面布局方式

CSS 的强大之处在于它不仅能控制网页中的元素样式，还能从整体上对 Web 页面进行布局。上一章介绍的浮动和定位就是 CSS 用来布局的核心技术。在过去，Web 设计人员使用 table 元素对页面进行布局，通过多个嵌套的表格来实现多行、多列的效果。但是这种做法违背了 (X)HTML 的语义性，是对 table 元素的滥用，因此是不可取的。

本章将介绍一些常见的网页布局方式，将说明它们的优势与不足各是什么，以及如何使用 CSS 来实现不同的布局方案。

本章主要内容

- 常见的布局形式
- 什么是固定式布局
- 什么是流动式布局
- 什么是弹性布局
- float 与 position 属性的对比

16.1　布局类型概述

页面的布局类型可大致分为以下 3 类：固定式布局、流动式布局和弹性布局。

(1)　固定式(Fixed)布局

采用固定式布局的页面，其宽度不跟随浏览器窗口的宽度进行变化，当然也不受显示器分辨率的影响。通常，固定式的页面内容宽度限定在 760px 左右，适合 800×600 的分辨率下的显示，随着 1024×768 分辨率的普及，不少页面也开始扩展固定宽度的大小。图 16-1 为网易的首页，它的宽度就固定在 750px，而中国雅虎的页面宽度为 950px，适合 1024×768 分辨率下的显示(见图 16-2)。

不管显示设备的尺寸如何，固定式布局的页面效果总是相同的。当浏览器窗口过小时，用户需要移动滚动条才能浏览整个页面，而当浏览器窗口过大时，页面会产生大量的空白区域。

(2)　流动式(Fluid)布局

采用流动式布局的页面，其宽度能自动适应浏览器的窗口大小。当用户调整浏览器窗口大小时，页面内容也会随之变化。这种布局方式不会在窗口中产生空白区域，空间利用率比较高，但是如果用户的显示设备过大或者过小，则页面效果可能会变得很差。卓越亚马逊就采用了流动式布局，页面左栏和中栏会自动适应不同宽度的浏览器窗口(见图 16-3)。

(3)　弹性(Elastic)布局

在采用弹性布局的页面中，元素会根据文字的大小进行变化。文字和页面元素使用百分数或者使用 em 作为单位的长度值，当用户通过浏览器调整文字大小时，页面元素会随着文字大小进行变化。英文雅虎的页面就是一个弹性布局的例子(如图 16-4 所示，图中左侧为正常文字

大小时页面效果，右侧为将文字大小设为较大时的效果)。

采用弹性布局的页面可以让用户在更改字体大小的同时，也能获得较好的效果。由于用户可以自行设置浏览器的默认字体大小值，页面效果也因此具有一定的不可预测性。



图 16-1　网易(http://www.163.com)采用固定宽度，宽度值固定在 750px



图 16-2　中国雅虎(http://cn.yahoo.com)采用固定宽度，宽度值固定在 950px



图 16-3　卓越亚马逊(http://www.amazon.cn)采用流动式布局



图 16-4　英文雅虎(http://www.yahoo.com)页面采用弹性布局，元素宽度会跟随文字大小进行变化

16.2　float 还是 position

上一章介绍了 CSS 的 float 和 position 属性，它们均可用来布局页面元素，那么对于这两种不同的布局方式，我们应该如何选择呢？它们的优势和劣势以及适用范围各是什么呢？

16.2.1　float 布局

float 属性使元素在原有的位置上向左或向右浮动。由于浮动元素的位置是基于其在文档流中的原始位置的，因此浮动布局的灵活性比较低。另外，许多浏览器的显示问题都是由 float 属性导致的，虽然大部分问题都有相应的解决方法，但这无疑会增加设计成本，降低开发速度。

float 属性一般用来实现比较简单的页面布局形式，且(X)HTML 元素和其布局位置相差不多。

16.2.2　position 布局

position 属性可将元素放置在浏览器窗口中的任何位置上，不受元素在原始文档流中位置



的干扰，因而具有较高的灵活性。但是已定位元素会脱离文档流，当元素大小不可知时，我们没有办法让其他元素相对于该元素进行定位(比如定位到该元素的底部)。

当元素顺序和其显示位置相差较多且元素大小固定时可选择定位属性进行布局。

16.3 布 局 实 战

在上一章曾介绍过如何使用浮动属性实现页面的布局，本章实战内容将利用 CSS 浮动原理分别实现 3 种不同类型的页面布局，我们将略去页面中的细节，只关注整体效果。上一章还介绍了如何用定位属性进行布局，考虑到实际开发中利用定位属性进行整体布局的情况比较少见，本章将不再讨论。

16.3.1 二分栏固定式布局

本小节将实现一个三行两列的固定式布局页面，其原理如图 16-5 所示。

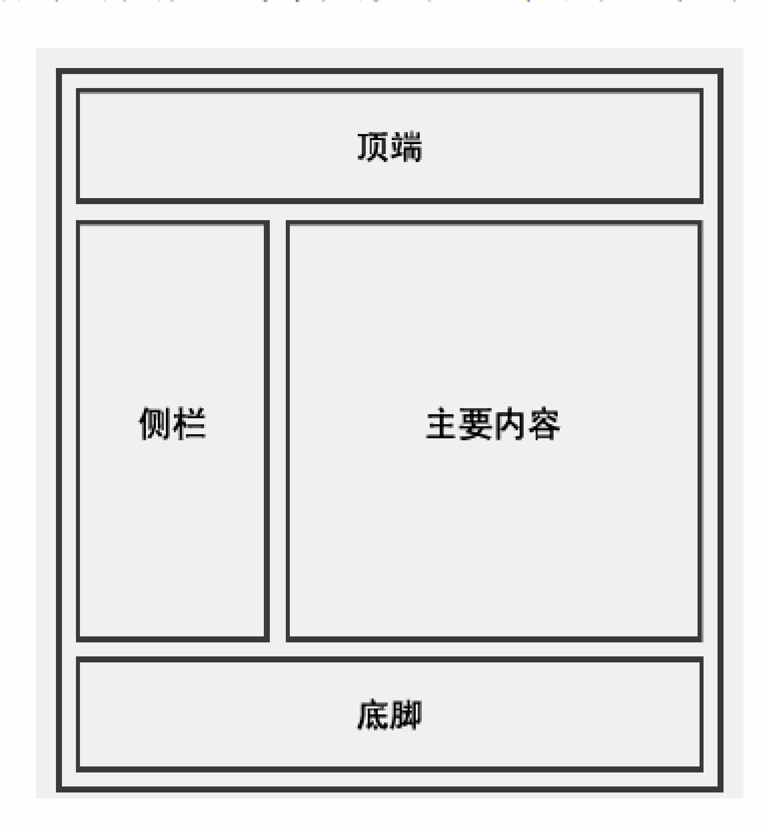


图 16-5 三行两列固定式布局原理

首先准备好(X)HTML 代码：

```
<div id="wrapper">
   <div id="header"></div>
   <div id="sidebar"></div>
   <div id="content"></div>
   <div id="footer"></div>
</div>
```

#wrapper 作为容器控制着整个页面的宽度，我们把该元素的宽度设为 760px：

```
div#wrapper{
    width:760px;
}
```

接着我们将侧栏向左浮动，并设置其宽度为 200px：



```
div#sidebar{
    float:left;
    width:200px;
}
```

主要内容部分可采用多种方式实现，这里我们让#content 也向左浮动：

```
div#content{
    float:left;
    width:550px;
    margin-left:10px;
}
```

最后我们给#footer 添加 clear 属性，使其能位于浮动元素的下方：

```
div#footer{
    clear:both;
}
```

以上为实现布局的基本 CSS 样式代码，为了获得更好的页面效果，我们修改或添加了一些样式，以下是完整的代码：

```
div#wrapper{
    width:760px;
    margin:auto;     /* 使整个页面水平居中 */
}
div#header{
    height:100px;
    margin-bottom:10px;
}
div#sidebar{
    float:left;
    width:198px;     /* 由于添加了 1px 宽的边框，所以宽度应设为 200px - 2px */
    height:300px;
    margin-bottom:10px;
}
div#content{
    float:left;
    width:548px;     /* 由于添加了 1px 宽的边框，所以宽度应设为 550px - 2px */
    height:400px;
    margin-left:10px;
    margin-bottom:10px;
}
div#footer{
    clear:both;
    height:80px;
}
.sectionStyle1{
    background:pink;
    border:1px solid gray;
```



```
}
.sectionStyle2{
    background:lightyellow;
    border:1px solid gray;
}

<div id="wrapper">
    <div id="header" class="sectionStyle1"></div>
   <div id="sidebar" class="sectionStyle2"></div>
   <div id="content" class="sectionStyle2"></div>
   <div id="footer" class="sectionStyle1"></div>
</div>
```

页面效果如图 16-6 所示。

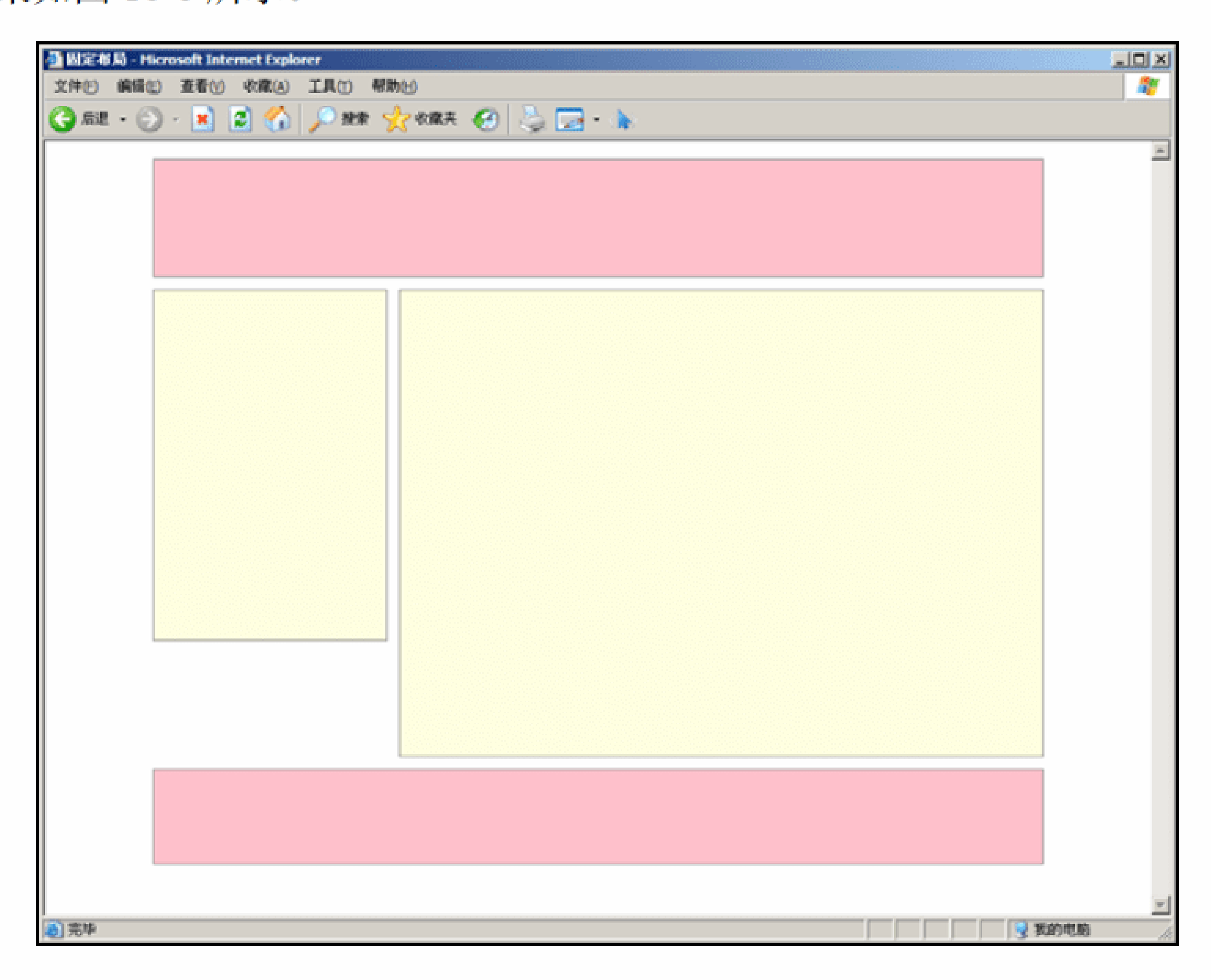


图 16-6　三行两列固定式布局效果

有时 Web 设计者对文档中元素出现的顺序要满足一定的要求，比如将主要内容放在导航、侧栏等一类的次要内容之前，以便当设备不支持 CSS 时，重要内容能位于次要内容之前。在本例中，假如要求#content 必须出现在#sidebar 之前，且还想保持现有的样式不变，该如何做呢？利用 CSS 中的负边距就可以轻松实现。

首先我们更改(X)HTML 文档，将#content 和#sidebar 的位置进行互换：

```
<div id="wrapper">
   <div id="header" class="sectionStyle1"></div>
   <div id="content" class="sectionStyle2"></div>
   <div id="sidebar" class="sectionStyle2"></div>
   <div id="footer" class="sectionStyle1"></div>
</div>
```



现在效果如图 16-7 所示(图为 Firefox 浏览器中的效果，IE6 中存在边距加倍问题，#sidebar 会因右侧空间不足而被挤到#content 下面)，根据文档流和浮动原理，#content 位于左侧，#sidebar 位于右侧，但这并不是我们想要的结果。

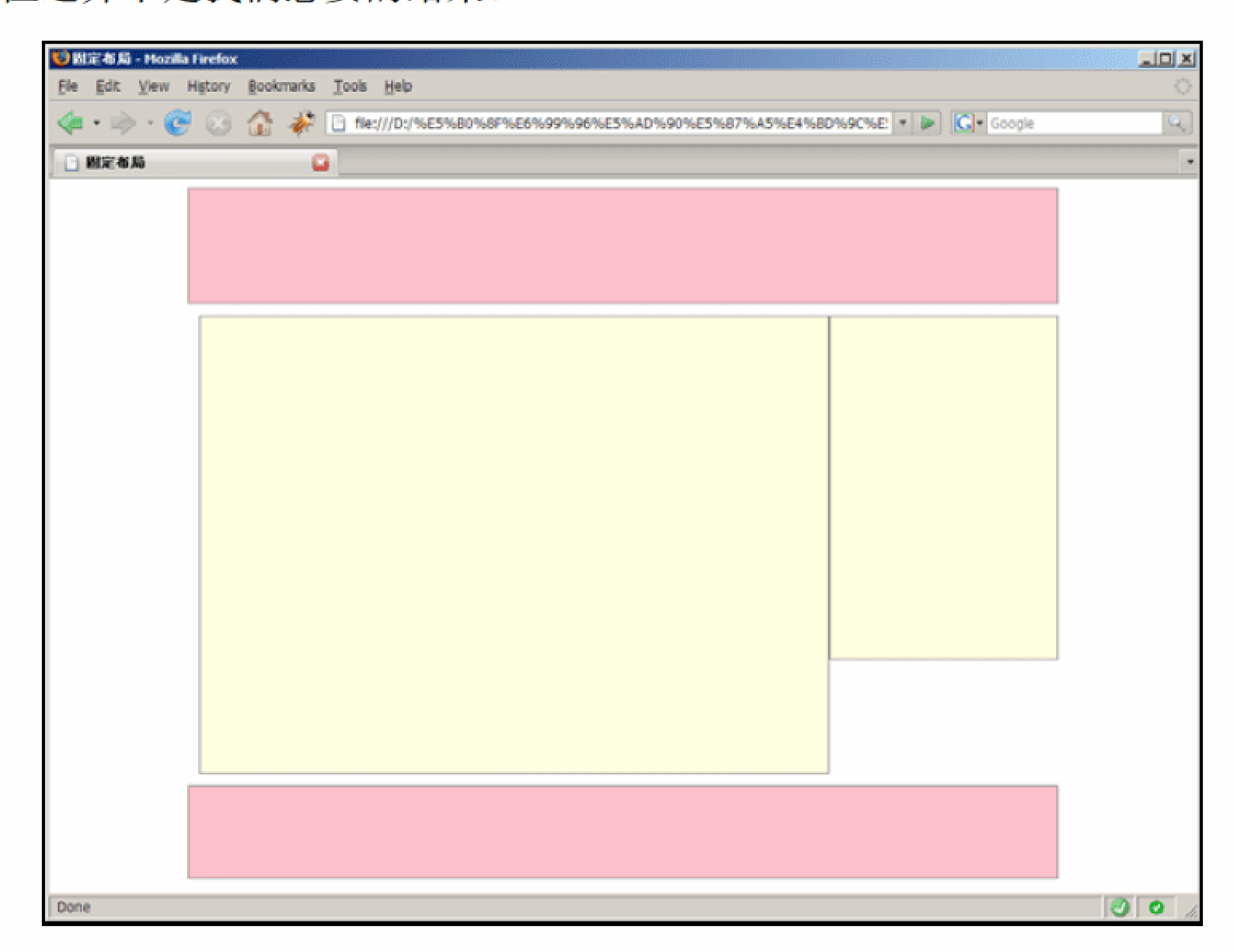


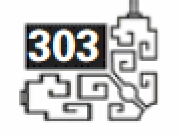
图 16-7　更改#sidebar 和#content 位置之后，二者左右位置颠倒(Firefox 浏览器)

我们的解决方案是给#content 增加足够的左边距，使左边有空间显示#sidebar，然后再给#sidebar 添加适当的负左边距，把它“拽”向左端。最后修正 IE6 的边距加倍的问题。#content 和#sidebar 的全部样式代码如下：

```
div#sidebar{
    float:left;
    width:198px;
    margin-bottom:10px;
    margin-left:-760px;     /* 将 sidebar“拽”向左端 */
    height:300px;
}
div#content{
    float:left;
    width:548px;
    margin-left:210px;      /* 留出左侧 sidebar 的空间 */
    margin-bottom:10px;
    height:400px;
    display:inline;         /* 修正 IE6 的边距加倍问题 */
}
```

效果最终如图 16-8 所示。

Element Fusion(http://www.elementfusion.com/)是一家网站设计、网络开发公司，该公司网站的页面就采用了左右两分栏的格局(见图 16-9)。



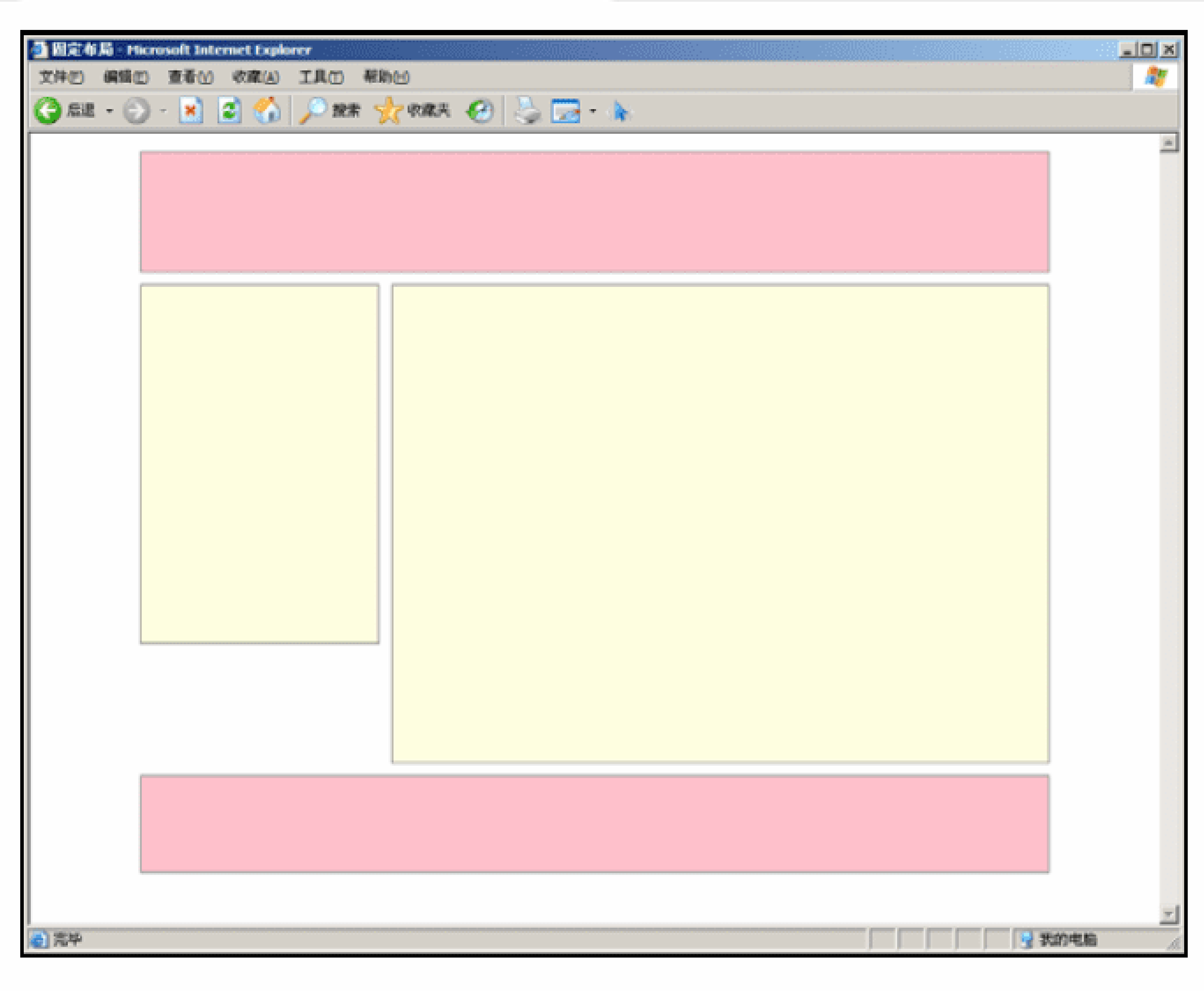


图 16-8　使用负边距实现布局

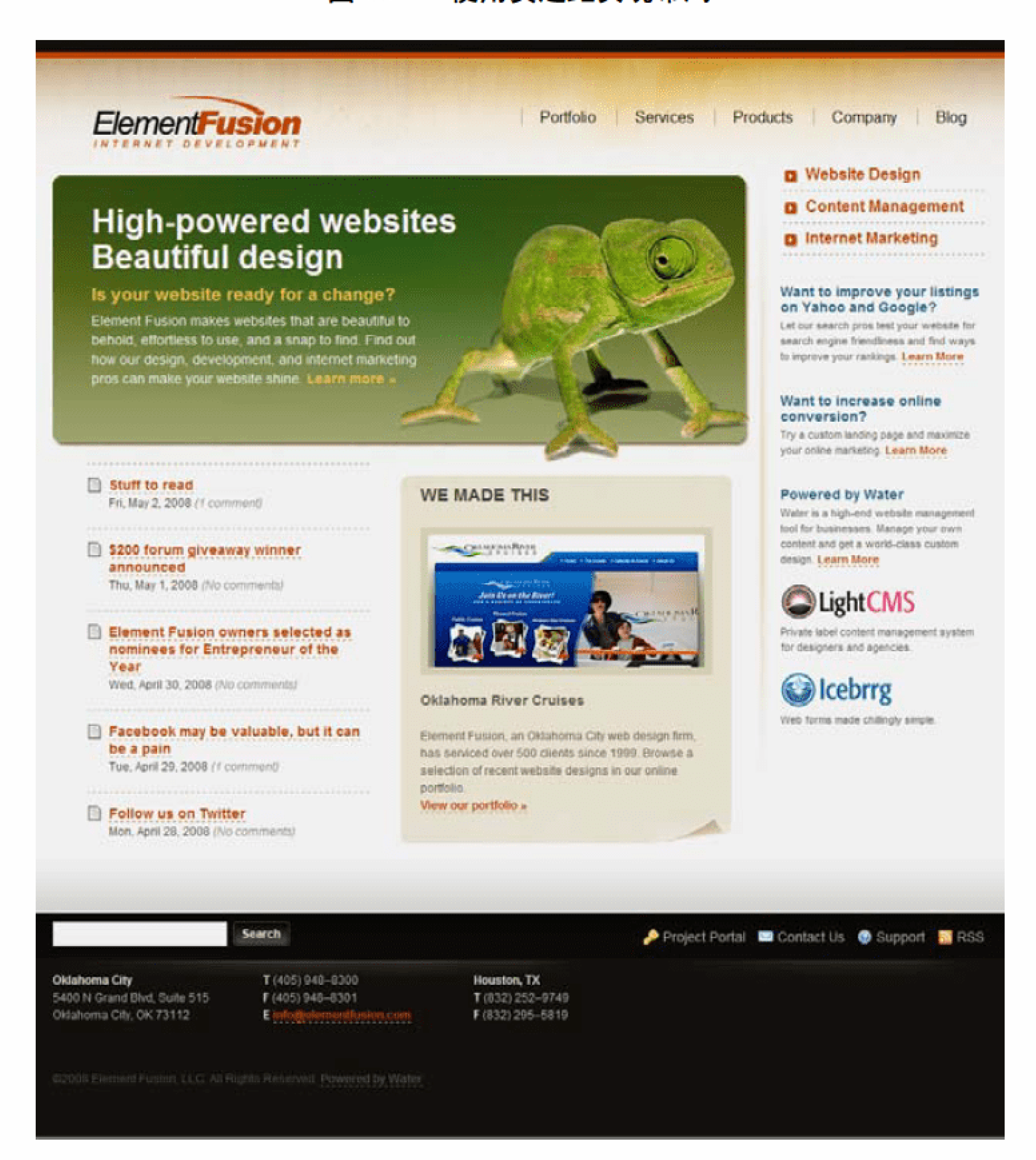


图 16-9　Element Fusion 网站页面采用了左右两栏的布局

页面内容部分的核心代码大致如下：

```
<div id="frame">
    <div id="header">...</div>
    <div id="container">...</div>
    <div id="extraCol">...</div>
</div>
<div id="footer">...</div>
```

包含左右两栏内容的容器#container 和#extraCol 分别指定了 float:left 和 float:right 声明，从而形成左右两栏的基本布局形式。为了使页面水平居中对齐，#frame 元素添加了 margin: 0px auto;声明。

提示：在确定左右两栏宽度时，一些设计师采用了黄金分割比例，以增强页面的美感。比如，若页面的整体宽度为 760px，那么主要内容区域的宽度就可定为 760/1.618=470px，侧栏宽度则为 760−470=290px。

16.3.2　三分栏流动式布局

流动式布局要求页面区域能自动适应浏览器的窗口，大小可灵活变化。其中的关键在于容器元素的宽度值单位不使用确定的长度值，而使用百分数或者不设置宽度属性。本小节首先将实现一个三行三列流动式的布局，其中中栏部分的宽度会自动变化，左右两个侧栏的宽度保持固定不变。图 16-10 为三行三列布局的示意图。

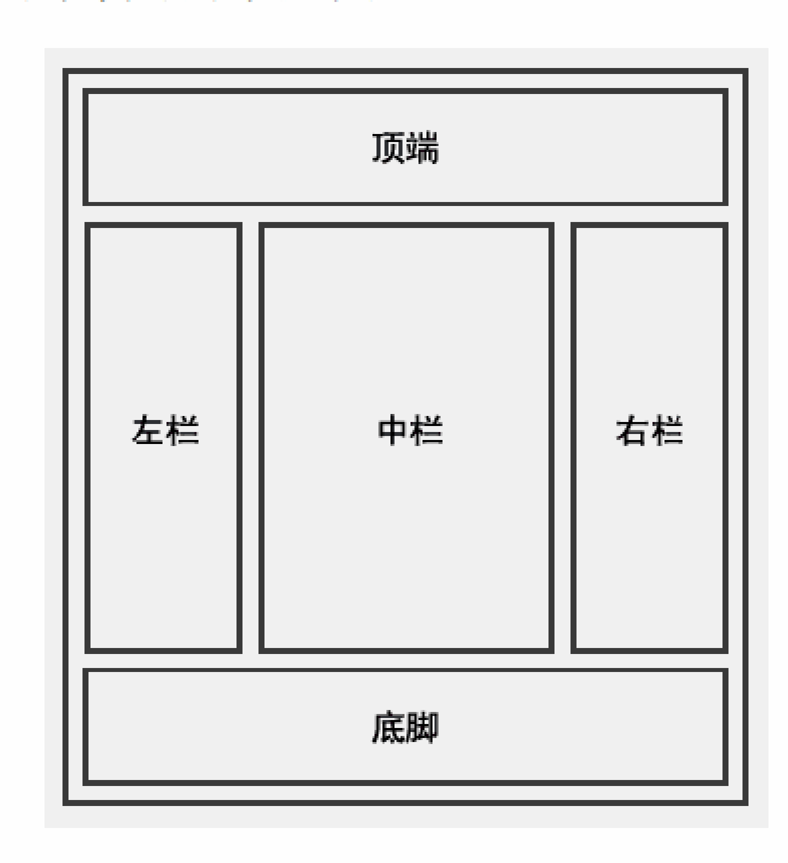
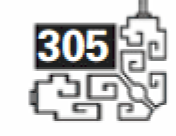


图 16-10　三行三列流动式布局

(X)HTML 代码如下：

```
<div id="header"></div>
<div id="sidebar"></div>
<div id="secondary"></div>
<div id="main"></div>
<div id="footer"></div>
```



我们让#sidebar 向左浮动，#secondary 向右浮动，最后给中栏设定适当的左右边距，完整的 CSS 代码和修改之后的(X)HTML 代码如下：

```
<style type="text/css">
div#header{
    height:100px;
    margin-bottom:10px;
}
div#sidebar{
    float:left;
    width:198px;
    margin-bottom:10px;
    height:300px;
}
div#secondary{
    float:right;
    width:198px;
    margin-bottom:10px;
    height:400px;
}
div#main{
    margin-left:210px;
    margin-right:210px;
    margin-bottom:10px;
    height:400px;
}
div#footer{
    clear:both;
    height:80px;
}
.sectionStyle1{
    background:pink;
    border:1px solid gray;
}
.sectionStyle2{
    background:lightyellow;
    border:1px solid gray;
}
</style>
<!--[if IE 6]>
<!-- 修正 IE6 浮动 bug -->
<style type="text/css">
div#main{
    margin-left:207px;
    margin-right:207px;
```



```
}
</style>
<![endif]-->

<body>
<div id="header" class="sectionStyle1"></div>
<div id="sidebar" class="sectionStyle2"></div>
<div id="secondary" class="sectionStyle2"></div>
<div id="main" class="sectionStyle2"></div>
<div id="footer" class="sectionStyle1"></div>
</body>
```

我们给左栏和右栏添加了固定的宽度值，中间一栏只设置了左右边距，因此它的宽度将按照块级元素的特性进行变化，即水平延展到容器边缘为止。代码最后添加了 IE6 特定的代码，其中把#main 的左右边距各减少了 3px，为什么这么做呢？前面说过 IE6 中非浮动元素没能完全占据浮动元素原来的位置，而是靠在浮动元素旁边。而 IE6 存在的 3px 间隔问题导致元素之间存在一定距离，因此 IE6 中#main 的左右边距应该减去这个边距值，这样区域之间的距离才真正是 10px。页面效果如图 16-11 所示。

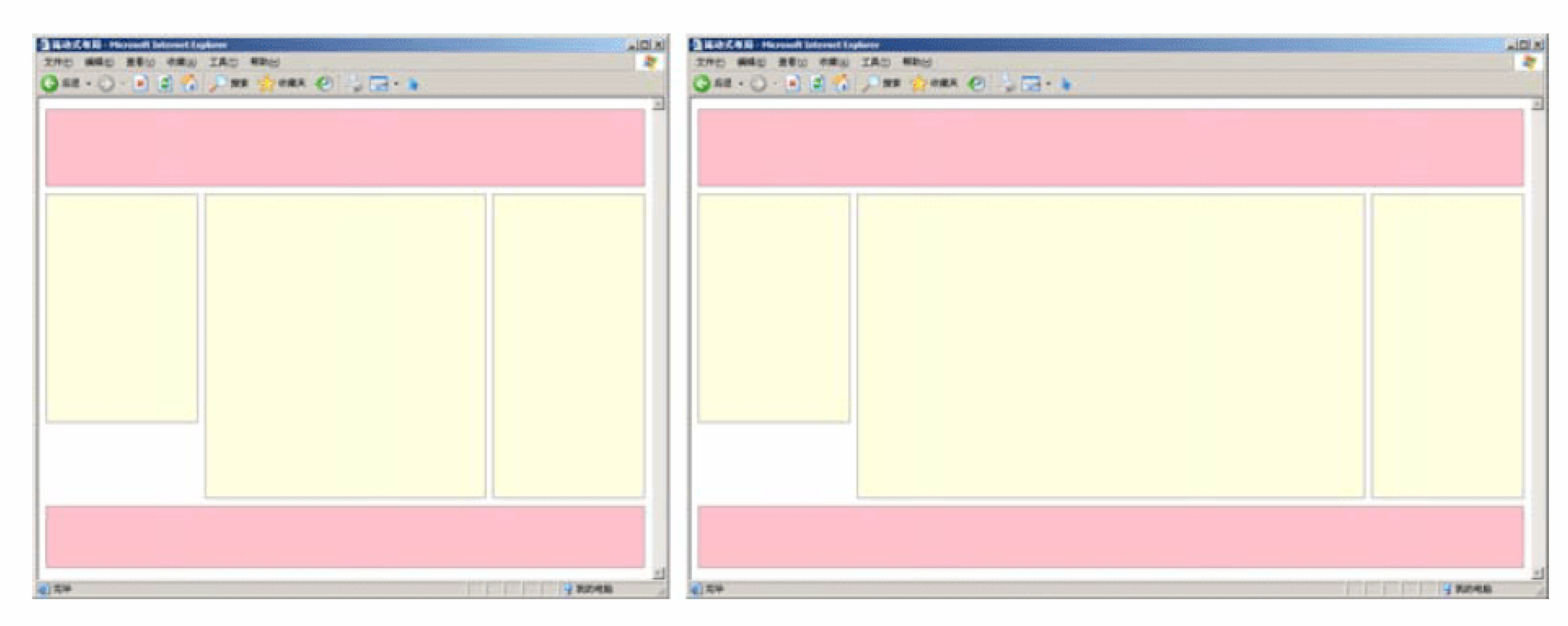
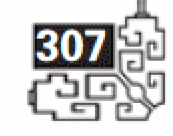


图 16-11　三栏流动式布局

当浏览器窗口宽度过小时，以上方式将会产生问题，左右浮动的元素会将中间一栏向下排挤，甚至右栏也会被排挤到左栏的下面。现在我们要对这种方式进行改进，防止窗口过窄导致页面布局混乱，同时我们将实现一种两栏流动、一栏固定的布局方式。

首先修改(X)HTML 代码，我们将原有的元素放置在一个 div 容器中，再将#sidebar 和#main 放在一个 div 容器中，代码如下：

```
<div id="wrapper">
    <div id="header" class="sectionStyle1"></div>
    <div id="secondary" class="sectionStyle2"></div>
    <div id="wrapperLeft">
        <div id="sidebar" class="sectionStyle2"></div>
        <div id="main" class="sectionStyle2"></div>
```



```
    </div>
    <div id="footer" class="sectionStyle1"></div>
</div>
```

包含左栏和中栏的容器#wrapperLeft 需要设置适当的右边距，让出右栏的空间。添加如下代码：

```
div#wrapperLeft{
    margin-right:210px;
    margin-bottom:10px;
}
```

在容器内，#sidebar 和#main 可全部向左浮动，或者一左一右浮动。左栏与中栏之间存在10px 间距，因此两列的宽度之和并不是 100%，余下空间留给了间距，否则会因容器的水平空间不足导致中栏移到左栏的下面。以下是对#sidebar 和#main 修改后的代码：

```
div#sidebar{
    float:left;
    width:19%;
    height:300px;
    margin-right:10px;
}
div#main{
    float:left;
    width:78.4%;
    height:420px;
}
```

由于我们要求当浏览器窗口宽度小于一定数值时，页面宽度应该保持不变。我们给最外侧的容器#wrapper 添加如下代码：

```
div#wrapper{
    min-width:750px;    /* 针对非 IE6 浏览器 */
    width:expression((document.documentElement.clientWidth < 770)?750:'auto');
    /* 针对 IE6 浏览器 */
}
```

min-width 属性设置#wrapper 宽度最小值为 750px，针对 IE6，使用 CSS 表达式实现相同的功能(读者可以参考本书第 17 章的相关内容)。表达式的含义是当浏览器可视区域的宽度小于770px 时，#wrapper 的宽度值就为 750px，否则为 auto。条件的边界值之所以设为 770px，是考虑了#wrapper 元素与浏览器窗口之间存在一定的空隙。

最后取消上例中针对 IE6 添加的样式代码，效果如图 16-12 所示。

从图 16-12 中可以看出#wrapperLeft 的下边距没有起作用，底脚和中栏之间没有空隙(空隙是否存在还与浏览器窗口宽度有关系)，这是因为#wrapperLeft 内的两个元素均是浮动元素，其高度则不会根据它们的高度进行变化，因此我们需要添加额外的元素来清除浮动。



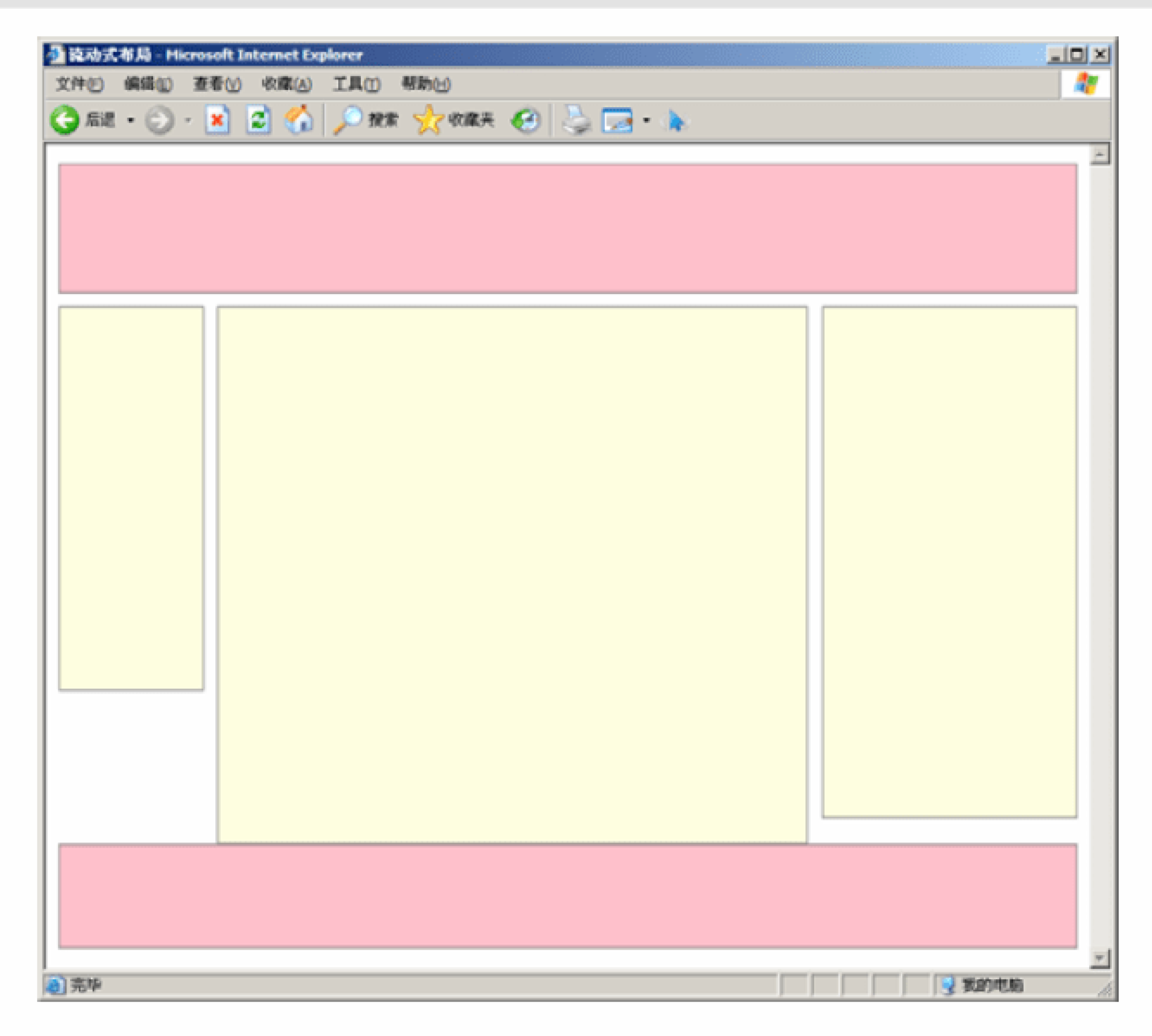
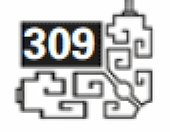


图 16-12　修改后的流动布局页面效果

完整代码如下：

```
div#wrapper{
    min-width:750px;
    width:expression((document.documentElement.clientWidth < 770)?750:'auto');
}
div#header{
    height:100px;
    margin-bottom:10px;
}
div#wrapperLeft{
    margin-right:210px;
    margin-bottom:10px;
}
div#sidebar{
    float:left;
    width:19%;
    height:300px;
    margin-right:10px;
}
div#secondary{
    float:right;
    width:198px;
    margin-bottom:10px;
    height:400px;
}
div#main{
    float:left;
```

```
    width:78.4%;
    height:420px;
}
div#footer{
    clear:both;
    height:80px;
}
div#hack{
    clear:both;
}
.sectionStyle1{
    background:pink;
    border:1px solid gray;
}
.sectionStyle2{
    background:lightyellow;
    border:1px solid gray;
}

<div id="wrapper">
   <div id="header" class="sectionStyle1"></div>
   <div id="secondary" class="sectionStyle2"></div>
   <div id="wrapperLeft">
      <div id="sidebar" class="sectionStyle2"></div>
      <div id="main" class="sectionStyle2"></div>
      <div id="hack"></div>
   </div>
   <div id="footer" class="sectionStyle1"></div>
</div>
```

效果如图 16-13 所示。从左图可以看出当窗口宽度过小时，页面宽度则保持在 750px，同时有横向滚动条产生。右图展示了当窗口变宽时，左栏和中栏都成比例地扩大，而右栏宽度保持不变。

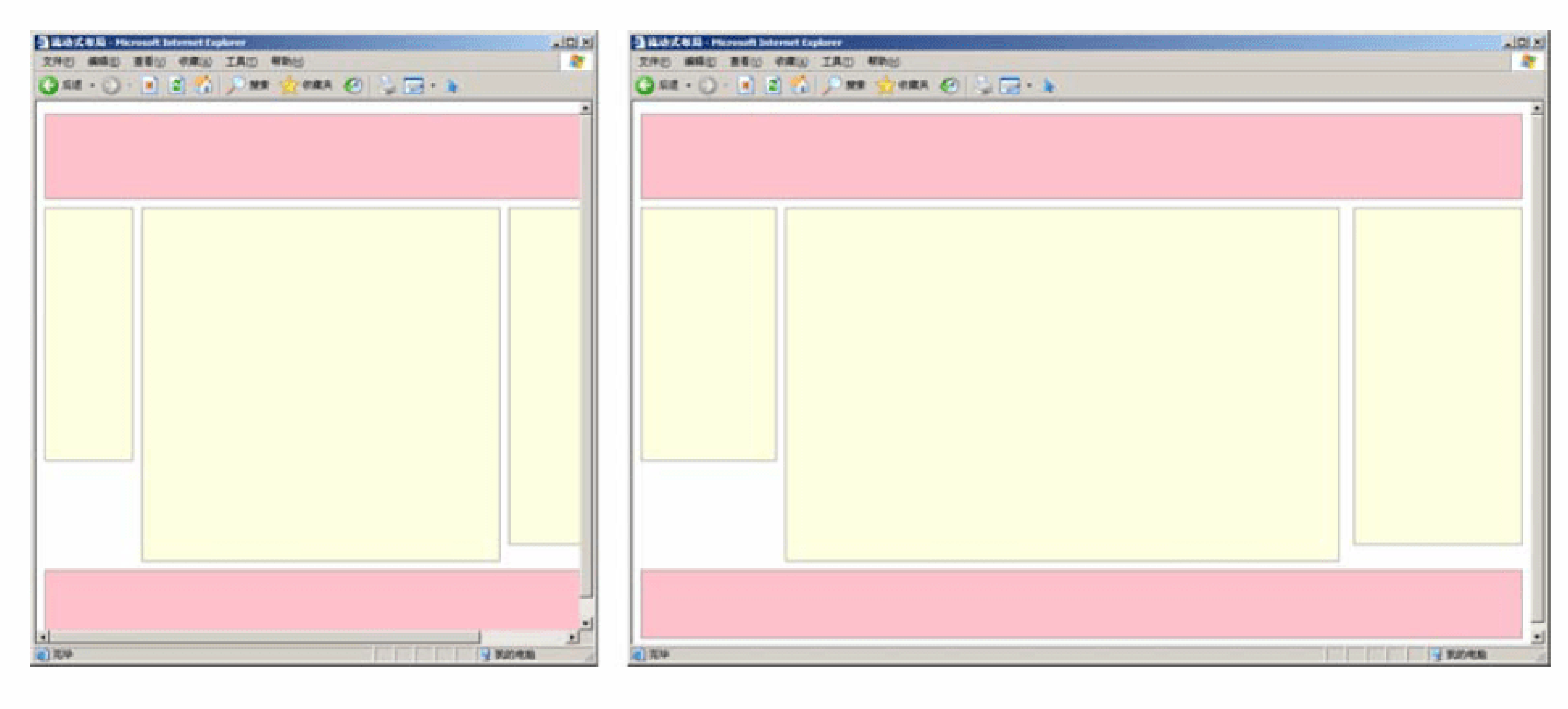


图 16-13　改进的三行三列流动式布局



不知道读者是否注意到，我们在前一例基础上修改(X)HTML 代码时，将#sidebar 和#secondary 进行了位置互换，本例中#secondary 位于#wrapperLeft 之前，那么当初能不能让#main 和#secondary 进行互换，使#secondary 位于其后呢。现在就来试一下，把(X)HTML 代码做如下修改：

```
<div id="wrapper">
    <div id="header" class="sectionStyle1"></div>
    <div id="wrapperLeft">
        <div id="sidebar" class="sectionStyle2"></div>
        <div id="main" class="sectionStyle2"></div>
        <div id="hack"></div>
    </div>
    <div id="secondary" class="sectionStyle2"></div>
    <div id="footer" class="sectionStyle1"></div>
</div>
```

页面效果如图 16-14 所示，#secondary 跑到下面去了。前面提到过，浮动元素脱离文档流，但是它的定位是基于文档流的，#secondary 元素向右浮动，但是它不能“高于”在文档流中的位置，应该位于#wrapperLeft 的下面。

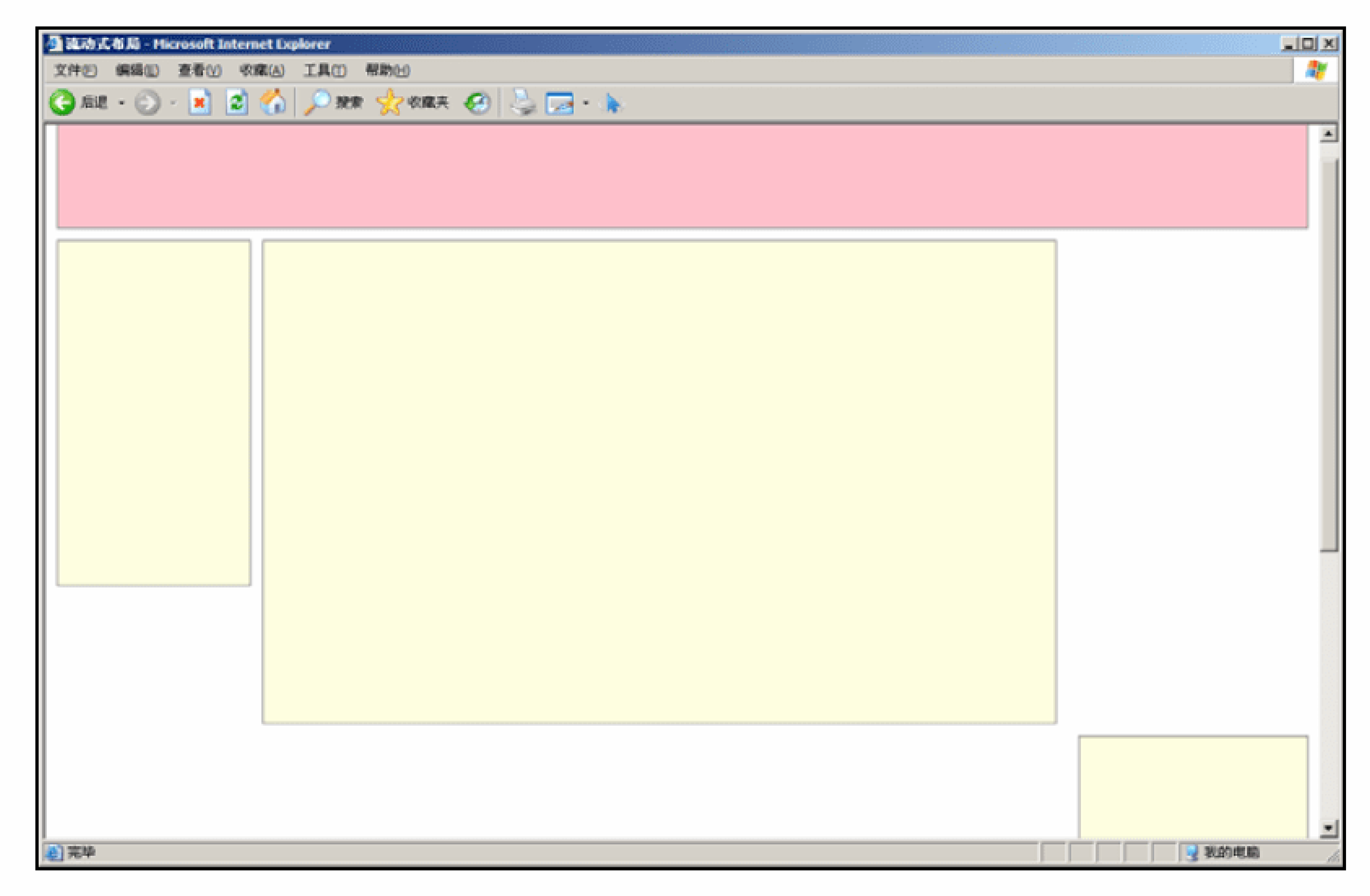


图 16-14　将#secondary 放置在#wrapperLeft 之后，它跑到下面去了

假如页面需要元素满足这样的顺序，我们又该如何让它正常显示呢？通过 CSS 的负边距可以解决这个问题。我们将#wrapperLeft 的样式做如下修改：

```
div#wrapperLeft{
    margin-right:-210px;
    margin-bottom:10px;
    float:left;
    width:100%;
}
```



现在效果如图 16-15 所示。

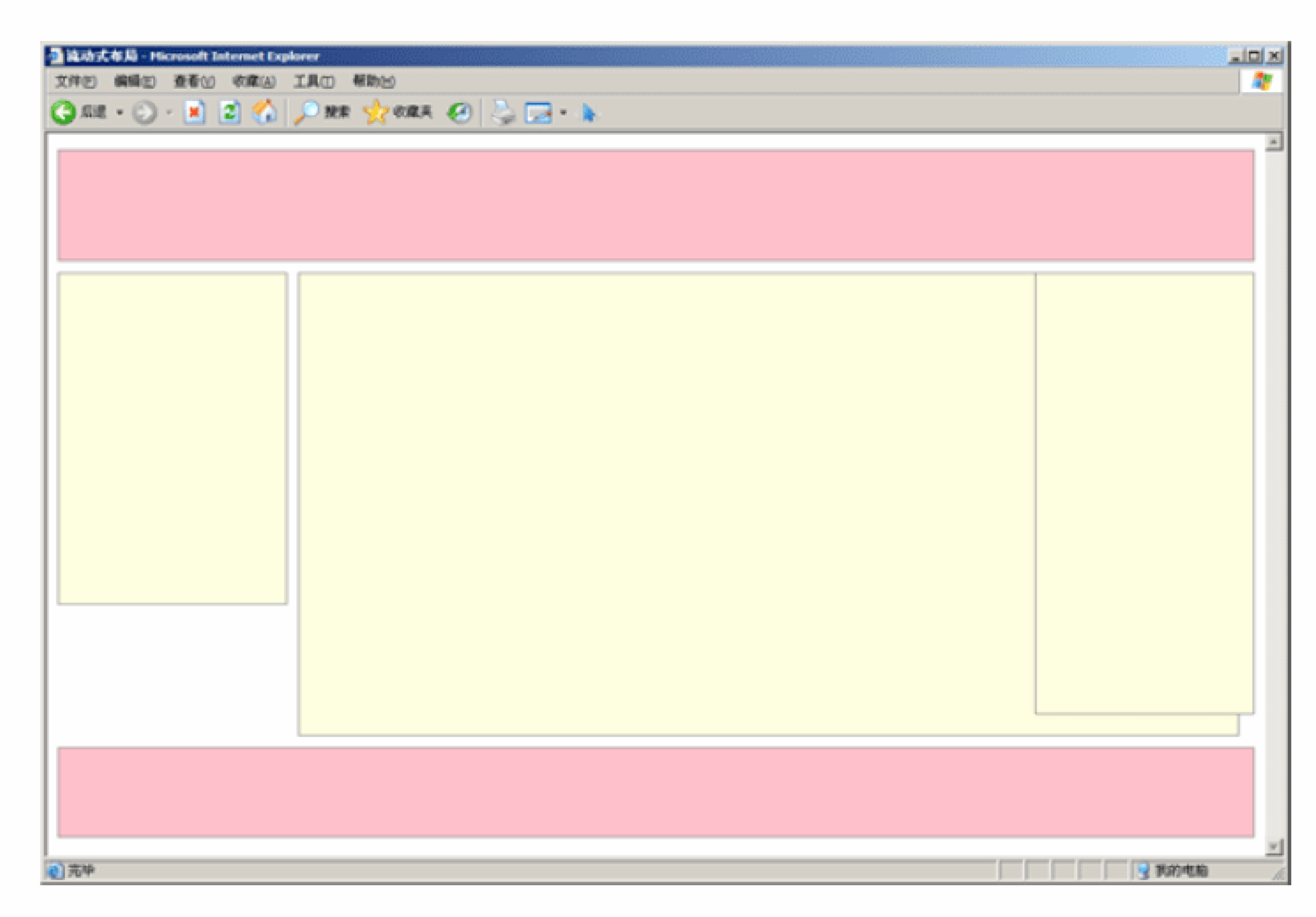


图 16-15　修改之后的页面效果

#wrapperLeft 的 margin-right 属性由 210px 变为-210px，我们给它设定了一个负边距，同时让它向左浮动，这使得在#wrapperLeft 的右侧空余出了 210px 宽的位置，正好能容下#secondary，因此#secondary 和#wrapperLeft 位于同一水平线上了。最后我们将#wraperLeft 中的元素的最右端向左移进 210px，我们将创建额外的 div 容器来处理这个问题。完整的代码如下：

```
div#wrapper{
    min-width:750px;
    width:expression((document.documentElement.clientWidth < 770)?750:'auto');
}
div#header{
    height:100px;
    margin-bottom:10px;
}
div#wrapperLeft{
    margin-right:-210px;
    margin-bottom:10px;
    float:left;
    width:100%;
}
div#contentLeft{
    margin-right:210px;
}
div#sidebar{
    float:left;
    width:19%;
    height:300px;
    margin-right:10px;
```

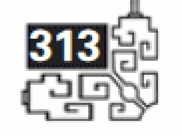


```
}
div#secondary{
    float:right;
    width:198px;
    margin-bottom:10px;
    height:400px;
}
div#main{
    float:left;
    width:78.4%;
    height:420px;
}
div#footer{
    clear:both;
    height:80px;
}
div#hack{
    clear:both;
}
.sectionStyle1{
    background:pink;
    border:1px solid gray;
}
.sectionStyle2{
    background:lightyellow;
    border:1px solid gray;
}

<div id="wrapper">
   <div id="header" class="sectionStyle1"></div>
   <div id="wrapperLeft">
    <div id="contentLeft">
          <div id="sidebar" class="sectionStyle2"></div>
          <div id="main" class="sectionStyle2"></div>
          <div id="hack"></div>
    </div>
   </div>
   <div id="secondary" class="sectionStyle2"></div>
   <div id="footer" class="sectionStyle1"></div>
</div>
```

页面效果如图 16-16 所示。

为了限制整个页面的宽度，我们给#wrapper 添加了 min-width 属性，但 IE 浏览器不支持这个属性，于是又针对 IE 使用了 CSS 表达式，表达式的内容是一行 JavaScript 代码，起的作用是当窗口宽度小于 770px 时，#wrapper 的宽度值取 750px，否则宽度值为 auto，即可自动适应浏览器窗口的宽度。本书第 17 章会介绍有关 CSS 表达式的内容。



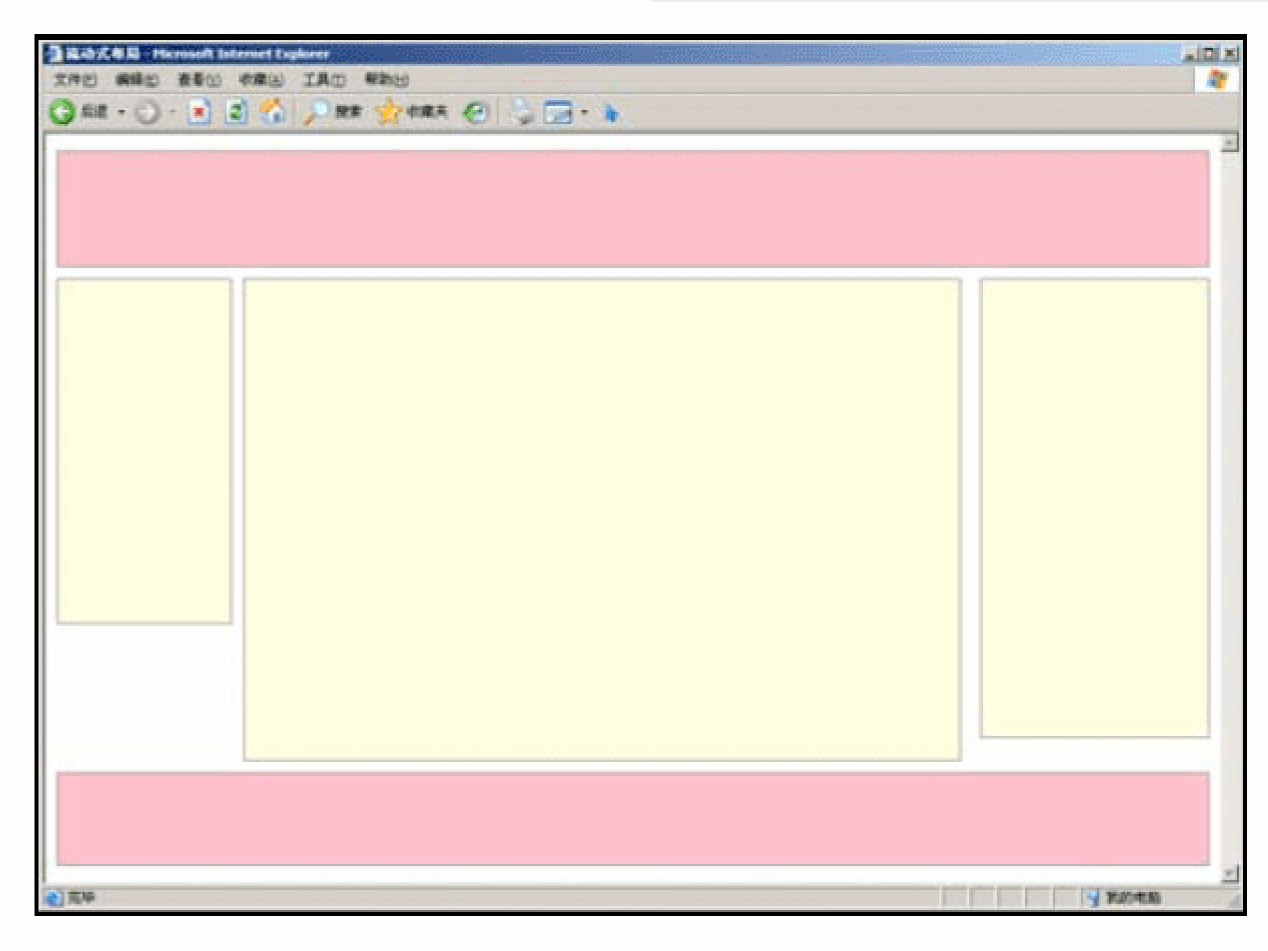


图 16-16　使用负边距方法完成页面布局

Biola University 的一个子站点(http://www.biola.edu/undergrad/)使用了三栏流动式布局，在一定范围内，页面中间的三个栏目的宽度均是可变的(见图 16-17)。

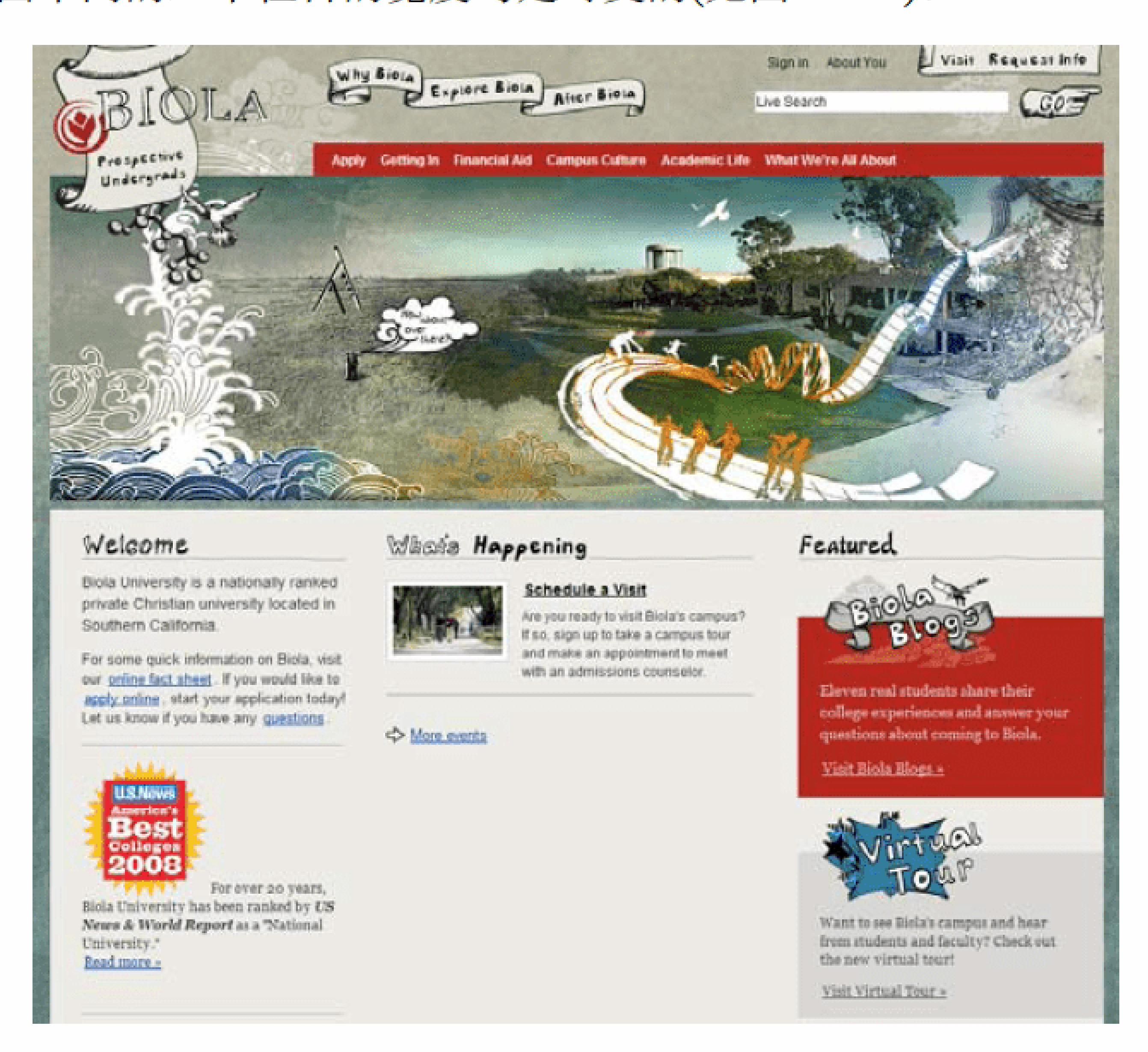


图 16-17　Biola University 网站的一个页面使用了流动式三分栏布局

表示三个栏目的 XHTML 代码如下：

```
<div id="content">
    <div id="homeWelcome">...</div>
    <div id="homeEvents">...</div>
    <div id="homeFeatures">...</div>
</div>
```



其中#homeWelcome 与#homeEvents 元素向左浮动，#homeFeatures 向右浮动，且三者的宽度值均是百分数。页面最外层的容器元素#wrapperPage 包含了如下样式，使得页面宽度变化限定在一定范围内：

```
max-width:1000px;
min-width:740px;
```

由于 IE 不支持这两个属性，设计者通过条件注释为 IE 添加了如下代码：

```
width:expression(document.body.clientWidth < 772 ? "740px" :
document.body.clientWidth > 1032 ? "1000px" : "100%" );
```

width 的属性值为一个 CSS 表达式(详见第 17 章)，它的含义是当浏览器可视区域宽度小于 772 个像素时，宽度值为 740px，可视区域宽度大于 1032px 时，宽度值取 1000px，其他情况下宽度值为 100%。

Vitamin(http://www.thinkvitamin.com/)网站在局部区域也运用了三栏式的布局形式(见图 16-18)，当浏览器宽度小于 1024 个像素的时候，三个栏目都会缩小自身的宽度以适应窗口的变化。读者可以自行分析网站的页面结构。

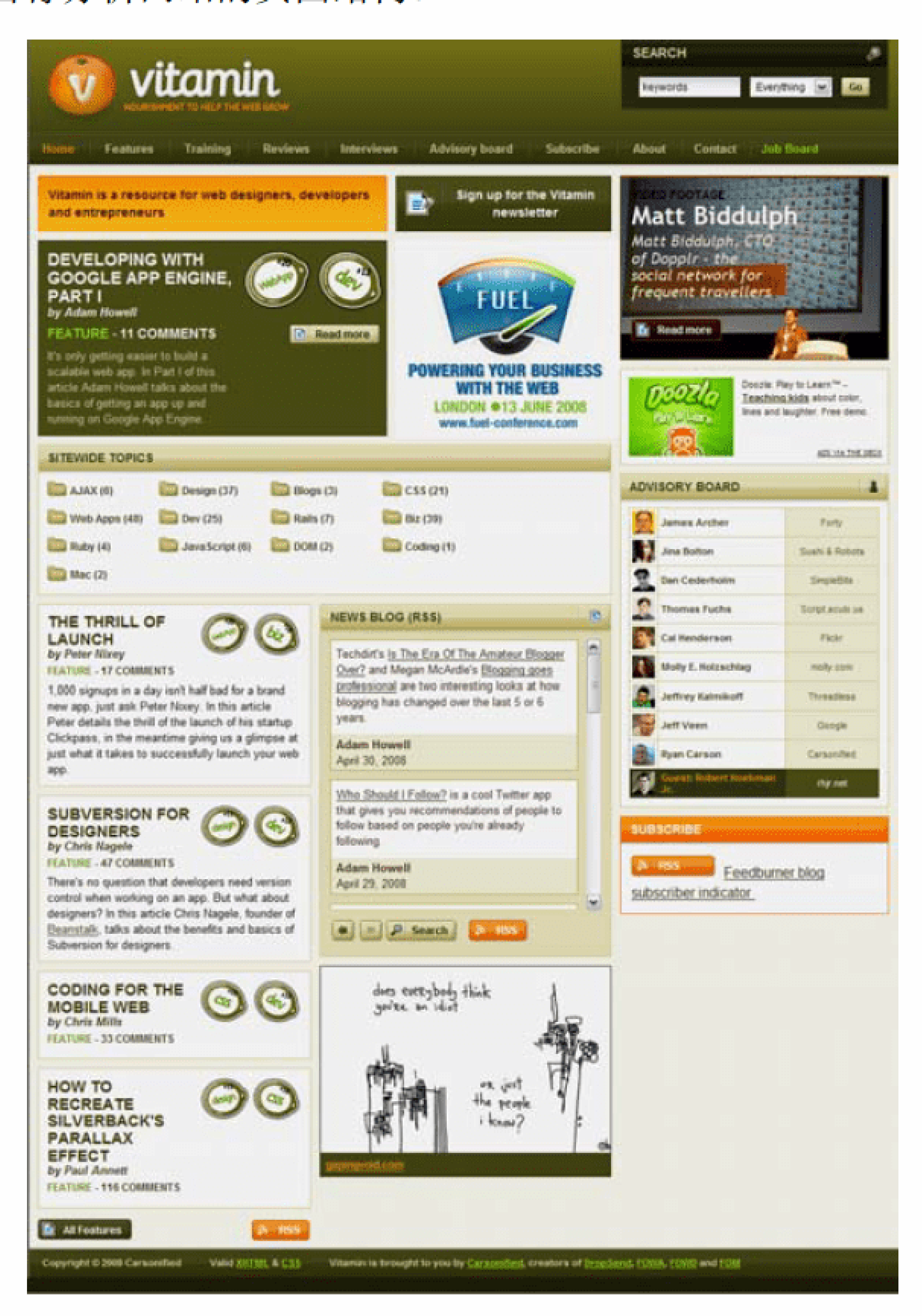
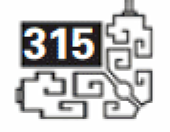


图 16-18　Vitamin 网站在局部使用了流动式三栏式布局



16.3.3 弹性布局

在国内，很少有网站采用弹性布局的方式，国内大部分用户也很少使用浏览器提供的字体大小调整功能。可调整的字体可以增强网页的可阅读性，为那些视力有障碍的人士提供方便，另外网站也能体现出较好的灵活性。弹性布局的关键在于，字体采用百分数或者以 em 为单位的长度值，元素的维度值单位均使用 em，当字体大小发生变化时，容器的宽度会跟随文字一起变化。

有些设计者对使用 em 作为单位来布局页面感到不适，他们更习惯于使用像素作为单位，觉得它能更精确地控制页面效果。下面我们来详细介绍字体大小与单位 em 和 px 之间的关系，相信读者在阅读完本小节内容之后就不再那么“惧怕”em 了。

第 5 章讲过，使用单位 em 的值是该元素 font-size 属性值的相对值。但是，当元素 font-size 属性值也采用 em 作为单位时，这个值就是其父元素 font-size 属性值的相对值了，这时它与百分数的作用是相同的。大多数浏览器默认的字体大小为 16px，所以下面的代码将 body 元素内的文字大小设为 16px：

```
body{font-size:1em;}
```

但是这种方式在 IE6 中会存在问题，当文字大小从正常调整为较大时，文字会一下子变得很大。我们使用百分数作为单位就可以避免这个问题：

```
body{font-size:100%;}
```

或者添加如下代码：

```
html{font-size:100%;}
```

设置好字体大小后，就要确定容器的宽度了，假如要创建一个宽度为 750px 的页面，我们需要将 px 单位转换为正确的 em 单位。由于 1em = 16px，则 1px = 1/16em。那么 750px 相当于 1/16×750=46.875em。

以下代码展示了弹性布局页面的基本实现方法：

```
html{
    font-size:100%;     /* 解决 IE6 中的字体变化程度过大的问题 */
}
body{
    font-size:0.75em;   /* 0.75 × 16px = 12px */
}
#wrapper{
    width:46.875em;
    margin:0px auto;
}
#header{
    height:7em;
    clear:both;
    margin-bottom:10px;
}
```



```
#sidebar{
    width:19%;
    height:25em;
    float:left;
}
#content{
    width:79%;
    height:25em;
    float:right;
}
#footer{
    margin-top:10px;
    height:6em;
    clear:both;
}
.sectionStyle1{
    background:pink;
}

<div id="wrapper">
    <div id="header" class="sectionStyle1">顶端</div>
   <div id="sidebar" class="sectionStyle1">侧栏</div>
    <div id="content" class="sectionStyle1">主要内容</div>
    <div id="footer" class="sectionStyle1">底脚</div>
</div>
```

效果如图 16-19 所示。左图为正常文字大小时的页面效果，右图为将文字设为较大后的效果。

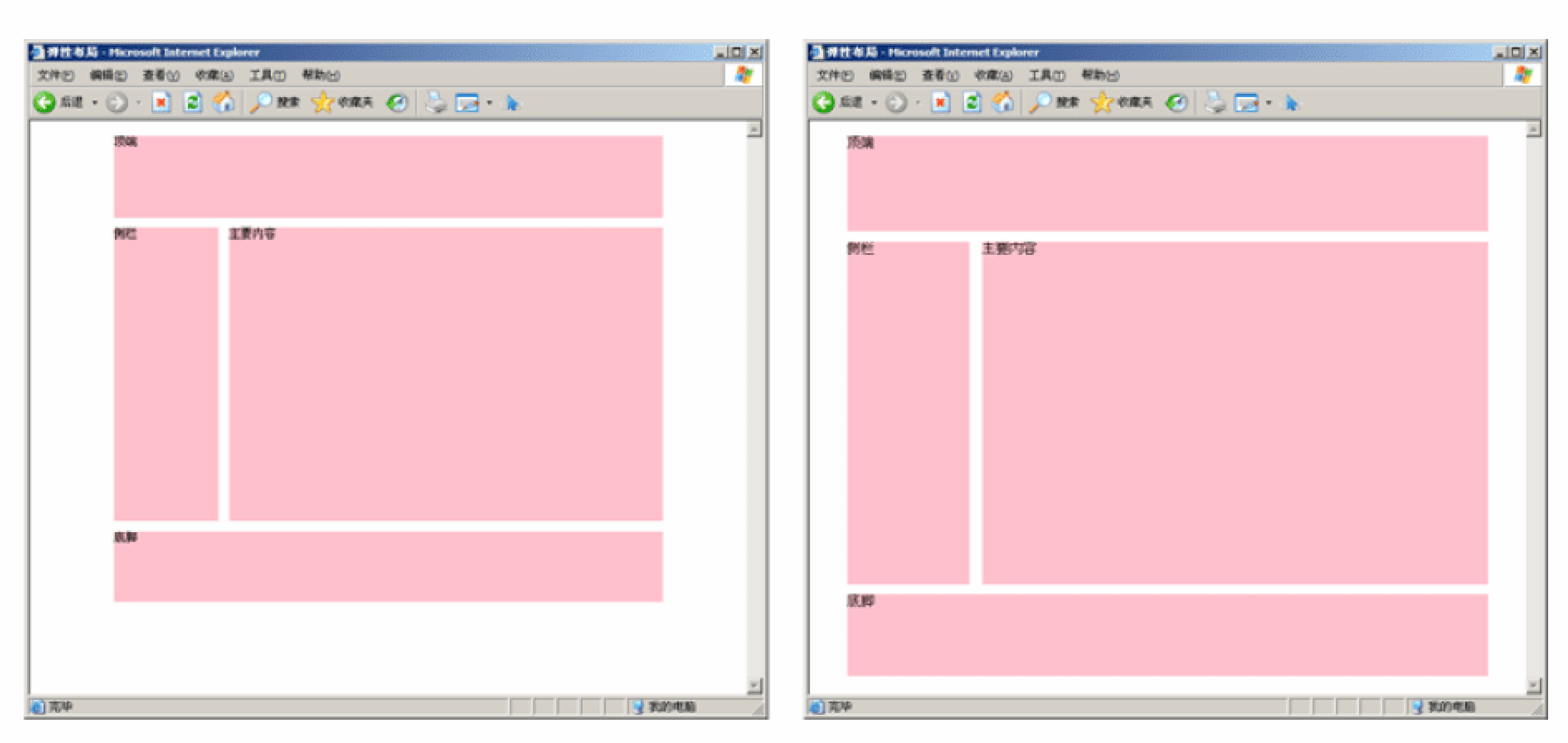


图 16-19　弹性布局的效果

CSS 禅意花园的一个效果(http://www.csszengarden.com/?cssfile=/063/063.css)中使用了弹性布局(见图 16-20)，页面文字的字号都是可以调节的，且元素会跟随文字一起变化。页面宽度设为 48em，其他一些元素的属性也都采用 em 作为单位。






图 16-20 “CSS 禅意花园”的一个效果中使用了弹性布局(左为正常字体，右为最大字体)

提示：为了使弹性布局页面更协调，有时也给 img 元素设置相对长宽值。根据图像原始大小计算出对应的采用 em 作为单位的值，然后通过 CSS 样式赋给特定的图像即可。注意换算单位时要使用 img 元素继承下来的字体大小值。美国在线(http://www.aol.com/)页面中的标识就是采用这种方式设计的。

16.4 小 结

本章介绍了网页中常用的三种布局方式：固定式、流动式和弹性布局。它们各有自己的优缺点。

固定式布局页面宽度固定不变，设计者可完全预期页面的效果。但是当用户显示设备过大时，窗口会出现大量的空白。

流动式布局的页面宽度随着浏览器窗口进行变化，不会产生空白，空间利用率高。但当浏览器过小或者过大时，页面效果反而会变差。在实际开发中经常将固定式和流动式结合在一起使用，当窗口宽度超出一定范围时，页面大小会保持固定。

弹性布局的页面元素宽度会随着文字大小进行变化，设计者需要将像素转换为 em 作单位的数值。弹性布局提高了页面的可访问性，为一些视力有障碍的人士提供了方便。但由于浏览器默认字体大小可更改，因此页面效果也会变得不可预测。

本章实战部分针对不同的布局方式提供了相应的解决方案，并介绍了负边距在布局中的使用，供读者参考和学习。

接下来将进入本书的第五部分——CSS 高级主题，内容涉及 CSS 在 XML、打印机、用户界面元素中的应用以及 IE 和 Firefox 对 CSS 的扩展。



第五篇　CSS 高级主题

第 17 章　CSS 高级应用

CSS 除了用于(X)HTML 外，还可以用于其他语言，XML 就是其中之一。

本章首先介绍有关 XML 的一些基本知识，以及如何创建一个 XML 文档(如果您已经对 XML 很熟悉，可以跳过此部分继续阅读)。接着介绍如何向 XML 文档添加样式、display 属性在控制 XML 样式中的作用以及如何对 XML 文档进行布局。

网页除了能通过浏览器显示给用户外，有时也会在打印机上进行输出。CSS 不仅能控制页面在显示器上如何显示，也能控制其打印输出的样式。本章将介绍如何使用 CSS 控制 Web 页面在打印机上输出的效果。

用户界面元素涉及了鼠标指针、系统颜色、系统字体以及轮廓线，本章将向读者一一介绍与这些内容相关的 CSS 样式属性。其中有关系统字体的内容已经在第 9 章中有过介绍，本章不再重复。

各浏览器厂商作为 CSS 规范的制定者，不断地在 CSS 中添加新的特性。在成为 CSS 正式规范之前，这些新特性都已经在一些浏览器中得到支持。这些新特性使 Web 设计人员能轻松实现各种特效。本章将介绍特定于 IE 与 Firefox 两大浏览器的 CSS 以及如何利用它们创造页面特效。

本章主要内容

- XML 的基本概念
- 如何给 XML 文档添加样式
- display 属性的作用
- 使用 CSS 布局 XML 文档中的元素
- 如何自定义图标
- 外轮廓线的使用
- 如何使用 CSS 滤镜解决 PNG 图像的透明问题
- Mozilla 的 CSS 扩展
- Mozilla 扩展中的属性和属性值

17.1 CSS 在 XML 中的应用

17.1.1 XML 概述

第 1 章讲到 XML 是一种用来创建其他语言的元语言，XHTML 就是由 XML 构造出来的。熟悉了 XHTML 之后就不难理解 XML 了。本节将会介绍一些有关 XML 的基础知识，读者若想详细了解 XML，请参考其他书籍。

1. 熟悉 XML 文档

首先来看如下这段代码：

```
<?xml version="1.0" encoding="utf-8"?>
<!-- some css books-->
<library>
    <book id="1">
        <title>Beginning CSS Web Development</title>
        <author>Simon Collison</author>
    </book>
    <book id="2">
        <title>CSS: The Missing Manual </title>
        <author>David Sawyer McFarland</author>
    </book>
    <book id="3">
        <title>CSS: The Definitive Guide</title>
        <author>Eric Meyer</author>
    </book>
    <book id="4">
        <title>CSS Cookbook</title>
        <author>Christopher Schmitt</author>
    </book>
</library>
```

首先，XML 文档由声明开始：

```
<?xml version="1.0" encoding="utf-8"?>
```

该声明指明 XML 的版本以及所使用的字符集。

接下来是一行注释，注释的语法与(X)HTML 相同：

```
<!-- some css books-->
```

标记<library>表明了文档的根元素，根元素中包含了几个相同的 book 元素，而每个 book 元素都包含了 title 和 author 两个元素：



```
<book id="1">
    <title>Beginning CSS Web Development</title>
    <author>Simon Collison</author>
</book>
```

book 元素有个 id 属性，title 和 author 元素中均包含了一些文本。

XML 文档中的元素名称、属性等都是自定义的，通过标记名称可以很容易理解文档的结构和内容。这是 XML 的优势之一，即使不使用任何注释也能明白它描述了一些书籍信息，而且很容易看出文档中各元素间的关系。

2. XML 文档的结构

XML 文档由两部分组成：序言(Prolog)和文档元素(即根元素)。序言包含有关该 XML 文档的一些信息，类似于(X)HTML 的 head 元素。例如上例的 XML 文档中就包含了一个 XML 声明和一行注释。文档根元素用来包含可能有的其他内容，且内容只能出现在文档根元素或根元素的内部。上例中的 library 就是根元素。

XML 中的元素、标记、属性等概念与(X)HTML 相似，这里不做过多的介绍。但需要注意以下几点：

- XML 中所有元素均要有起始标记和结束标记，元素需正确嵌套。
- XML 中只能包含一个根元素。
- XML 区分大小写。

在本章后面的 XML 示例中，只给出文档的根元素，会省略如下 XML 声明：

```
<?xml version="1.0" encoding="gb2312"?>
```

读者在练习的时候请注意，如果不声明相应的字符集，中文内容将无法正常显示或者导致 XML 解析失败。

17.1.2 使用 CSS

1. XML 的显示

(X)HTML 中每个元素的含义以及应该以何种方式显示，浏览器是知道的，因此不同元素会以特定的方式显示出来。XML 则不同，元素完全是自己定义的，浏览器不知道该如何处理这些陌生的元素。

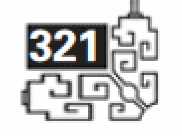
图 17-1 为浏览器对 17.1.1 小节中 XML 文档的显示。除了元素内容外，XML 声明、注释、表示元素的标记和元素中属性都被原封不动地显示出来。

2. 添加样式表

XML 文档只能使用外部样式表文件，在 XML 声明之后插入以下代码就可以为 XML 文档添加样式表：

```
<?xml-stylesheet type="text/css" href="styles.css" ?>
```

type 属性表明这是一个 CSS 文本文档，href 属性指明了文档位置。现在把该代码添加到



17.1.1 小节的 XML 文档中，再看一下浏览器的显示效果(见图 17-2)，即使没有样式文件，浏览器的显示效果也发生了改变。

```
<?xml version="1.0" encoding="utf-8" ?>
<!--  some css books  -->
- <library>
  - <book id="1">
      <title>Beginning CSS Web Development</title>
      <author>Simon Collison</author>
    </book>
  - <book id="2">
      <title>CSS: The Missing Manual</title>
      <author>David Sawyer McFarland</author>
    </book>
  - <book id="3">
      <title>CSS: The Definitive Guide</title>
      <author>Eric Meyer</author>
    </book>
  - <book id="4">
      <title>CSS Cookbook</title>
      <author>Christopher Schmitt</author>
    </book>
  </library>
```

图 17-1　浏览器对 XML 文档的显示

```
Beginning CSS Web Development Simon Collison CSS: The Missing Manual David
Sawyer McFarland CSS: The Definitive Guide Eric Meyer CSS Cookbook
Christopher Schmitt
```

图 17-2　添加引用样式表文件的代码后，浏览器的显示效果

在后面的范例中，引用样式表文件的代码会省略，读者在实际练习时请自行添加样式表文件，并引用到相应的 XML 文档中。

3. 选择符

CSS 通过选择符可以控制(X)HTML 文档中特定元素的样式，这同样适用于 XML 文档，但需要注意的是 id 选择符和 class 选择符不会对 XML 产生效果，浏览器并不知道 XML 文档元素中的 id 属性和 class 属性究竟是什么含义，因为 XML 文档中属性的含义同元素名称一样，完全是自己定义的。除非使用文档类型声明(DTD)或者 XML 模式(XML Schema)将 id 属性与元素的 id 对应上。

17.1.3　为 XML 文档添加样式

1. display 属性

使用 display 属性可以让 XML 元素以某种特定的方式显示出来，比如属性值 block 就可以让 XML 元素以(X)HTML 中块级元素的显示方式显示。

2. 使用 display:block 和 display:inline 设定元素的基本显示方式

使用 display:block 或 display:inline 就能使 XML 元素以(X)HTML 中的块级元素或内联元素

方式显示，如同 div 和 span 元素一样。请看示例：

```
<paragraph>
    这是一个段落，里面包含了一些<important>重要</important>的信息。
</paragraph>
```

我们为其指定如下样式：

```
paragraph{
    display:block;
    padding:10px;
    border:1px solid gray;
}
important{
    display:inline;
    color:red;
    font-weight:bold;
}
```

paragraph 元素和 important 元素的 display 属性分别设为 block 和 inline，图 17-3 为显示效果。它们和(X)HTML 中的 p 元素和 em 或 strong 元素的显示效果是一致的。

这是一个段落，里面包含了一些**重要**的信息。

图 17-3　使用 block 和 inline 属性值

3. 使用格式化表格数据

表格式的数据很容易在(X)HTML 文档中显示，使用 table、tr、td 等元素就可以在浏览器中显示为表格。那么我们如何处理 XML 文档中的表格式数据呢？display 属性的某些属性值可以做到，它们专门用来布局表格数据，使元素以某种表格元素的样式显示。

表 17-1 总结了 display 属性中与表格相关的属性值以及所代表的含义。

表 17-1　与表格相关的 display 属性值及其含义

属 性 值	含　义
table	定义一个块级的表格，相当于(X)HTML 的 table 元素
inline-table	定义一个内联级的表格
table-row	定义表格中的一行，相当于(X)HTML 的 tr 元素
table-row-group	定义包含一个或多个行的组，相当于(X)HTML 的 tbody 元素
table-header-group	定义包含一个或多个行的组，这些行位于表格的头部，相当于(X)HTML 的 thead 元素
table-footer-group	定义包含一个或多个行的组，这些行位于表格的头部，相当于(X)HTML 的 tfoot 元素
table-column	定义一列单元格，相当于(X)HTML 的 col 元素
table-column-group	定义包含一个或多个列的组，相当于(X)HTML 的 colgroup 元素
table-cell	定义表格中的一个单元格，相当于(X)HTML 的 td 元素
table-caption	定义表格标题



了解了这些属性值的含义后，我们就可以对包含表格式数据的 XML 文档添加样式了，请看示例：

```
<document>
    <table>
        <tableHead>
            <tableRow>
                <tableCell>日期</tableCell>
                <tableCell>交通工具</tableCell>
                <tableCell>行程</tableCell>
            </tableRow>
        </tableHead>
        <tableBody>
            <tableRow>
                <tableCell>3 月 1 日</tableCell>
                <tableCell>飞机</tableCell>
                <tableCell>北京 - 罗马</tableCell>
            </tableRow>
            <tableRow>
                <tableCell>3 月 2 日</tableCell>
                <tableCell>汽车</tableCell>
                <tableCell>罗马 - 蒙特卡帝尼</tableCell>
            </tableRow>
            ...
        </tableBody>
    </table>
</document>
```

添加的样式代码如下：

```
document{
    display:block;
    margin:10px;
}
table{
    display:table;
    border:1px solid #565656;
    border-collapse:collapse;
    font-size:12px;
    width:300px;
}
tableHead{
    display:table-header-group;
    font-weight:bold;
    background:#DFDFDF;
}
tableBody{
    display:table-row-group;
}
tableRow{
```



```
    display:table-row;
}
tableCell{
    display:table-cell;
    border:1px solid #565656;
    padding:6px;
}
```

除了给每个元素指定 display 属性之外，它与给(X)HTML 文档添加样式没有区别。如图 17-4 所示为 Firefox 浏览器的显示效果，它看上去与(X)HTML 的表格没什么两样。但遗憾的是 IE 浏览器不支持 display 属性中表格相关的属性，因此 IE 的显示比较混乱。

日期	交通工具	行程
3月1日	飞机	北京 - 罗马
3月2日	汽车	罗马 - 蒙特卡帝尼
3月3日	汽车	蒙特卡帝尼 - 威尼斯
3月4日	汽车	威尼斯 - 茵斯不卢克 - 慕尼黑
3月5日	汽车	慕尼黑 - 法兰克服
3月6日	汽车	法兰克服 - 琉森
3月7日	汽车	琉森 - 日内瓦
3月8日	汽车	日内瓦 - 卢森堡
3月9日	汽车	卢森堡 - 巴黎
3月11日	飞机	巴黎 - 北京

图 17-4　使用 display 表格相关的属性格式化表格数据(Firefox)

4. 使用 display:list-item 格式化列表

最后一个 display 属性就是 list-item 了，它将元素指定为一个列表项。CSS 与列表相关的属性都可以使用。请看示例：

```
<document>
<menu>
    <coffeeMenu>
        <menuTitle>咖啡类</menuTitle>
        <menuItem>玛塔里摩卡</menuItem>
        <menuItem>意大利浓缩</menuItem>
        <menuItem>牙买加蓝山</menuItem>
    </coffeeMenu>
    <teaMenu>
        <menuTitle>茶类</menuTitle>
        <menuItem>西湖龙井</menuItem>
        <menuItem>茉莉花茶</menuItem>
        <menuItem>普洱茶</menuItem>
    </teaMenu>
</menu>
</document>
```



现在添加如下样式：

```
document{
    display:block;
    margin:10px;
}
menu, coffeeMenu, teaMenu,  menuTitle{
    display:block;
}
menuItem{
    display:list-item;
}
```

item 元素的 display 属性设为了 list-item，item 元素则按照列表方式显示，浏览器会给(X)HTML 的 li 元素添加项目符号，对于该 item 元素也是如此(见图 17-5)。

咖啡类
- 玛塔里摩卡
- 意大利浓缩
- 牙买加蓝山

茶类
- 西湖龙井
- 茉莉花茶
- 普洱茶

图 17-5　使用 display:list-item 属性

再次增加如下样式，进一步美化这个列表：

```
menu{
    background:#F7EDD2;
    border:1px solid #9A7452;
    width:120px;
    padding:8px;
    color:#633A15;
    font-size:12px;
}
coffeeMenu, teaMenu{
    margin-bottom:6px;
}
menuTitle{
    font-weight:bold;
}
menuItem{
    line-height:1.6em;
    list-style-position:inside;
}
```

最终效果如图 17-6 所示。



咖啡类

- 玛塔里摩卡
- 意大利浓缩
- 牙买加蓝山

茶类

- 西湖龙井
- 茉莉花茶
- 普洱茶

图 17-6　最终效果

17.1.4　CSS 布局

使用 CSS 给 XML 元素进行布局与布局 XHTML 的原理是一样的，我们举几个简单的示例来说明如何使用 CSS 布局 XML 元素。

1. 默认布局

在未添加任何样式之前，XML 采用默认布局方式。这种状态下，块级元素会由上至下依次排列，内联元素会从左向右依次排列，且当水平空间不足时产生换行。

2. 浮动

当设定元素的 float 属性时，该元素将采取浮动定位，其他元素会在其周围排列。例如：

```
<document>
    <notice>这是一个浮动的元素。</notice>
    <paragraph>
    使用 CSS 的 float 属性可以将元素浮动起来，非浮动元素中的内容会排列在浮动元素的周围。
    采用浮动定位的优势之一在于，当浮动元素在 XML 文档中的位置发生变化时，通过调整 CSS
    代码，页面效果仍然可以保持一致。
    </paragraph>
</document>
```

样式代码：

```
document{
    display:block;
    margin:10px;
}
paragraph{
    float:left;
    display:block;
    width:500px;
    padding:10px;
    border:1px solid gray;
}
notice{
    float:left;
```

```
    width:100px;
    padding:10px;
    border:1px solid gray;
}
```

以上代码在 IE 浏览器中效果如图 17-7 所示。

这是一个浮动的元素。

使用CSS的float属性可以将元素浮动起来，非浮动元素中的内容会排列在浮动元素的周围。采用浮动定位的优势之一在于，当浮动元素在XML文档中的位置发生变化时，通过调整CSS代码，页面效果仍然可以保持一致。

图 17-7　浮动 XML 元素

3. 相对定位

相对定位只是在正常定位的基础之上增加一些变化。元素首先依照正常定位方式排列，然后根据指定的值进行偏移。请看示例：

```
<document>
    <paragraph>这是一个段落，里面包含了一些文字，<bias>有些</bias>文字的位置有偏移。
    </paragraph>
</document>
```

CSS 样式为：

```
document{
    display:block;
    margin:0
}
paragraph{
    display:block;
    border:1px solid gray;
    padding:10px;
}
bias{
    display:inline;
    position:relative;
    bottom:6px;
    font-weight:bold;
}
```

bias 元素采用相对定位，并设置 bottom 为 6px，这表示该元素会向上偏移 6 个像素。效果如图 17-8 所示。

这是一个段落，里面包含了一些文字，**有些**文字的位置有偏移。

图 17-8　相对定位

4. 绝对定位

绝对定位的元素将彻底从文档的正常排列之中分离开。它的位置基于已经定位的祖先元素(即具有非默认定位方式的最近的父元素)，偏移值通过 left、top、right 和 bottom 四个属性来确定。请看示例：

```
<document>
    <container>
        <box></box>
    </container>
</document>
```

样式代码：

```
container{
    position:absolute;
    left:100px;
    top:50px;
    height:200px;
    width:300px;
    border:1px solid gray;
    background:lime;
}
box{
    position:absolute;
    top:10px;
    left:20px;
    height:100px;
    width:100px;
    border:1px solid gray;
    background:yellow;
}
```

图 17-9 为 Firefox 的显示效果。container 元素以浏览器窗口为基准产生位移，而 box 元素则相对其父元素 container 进行定位。

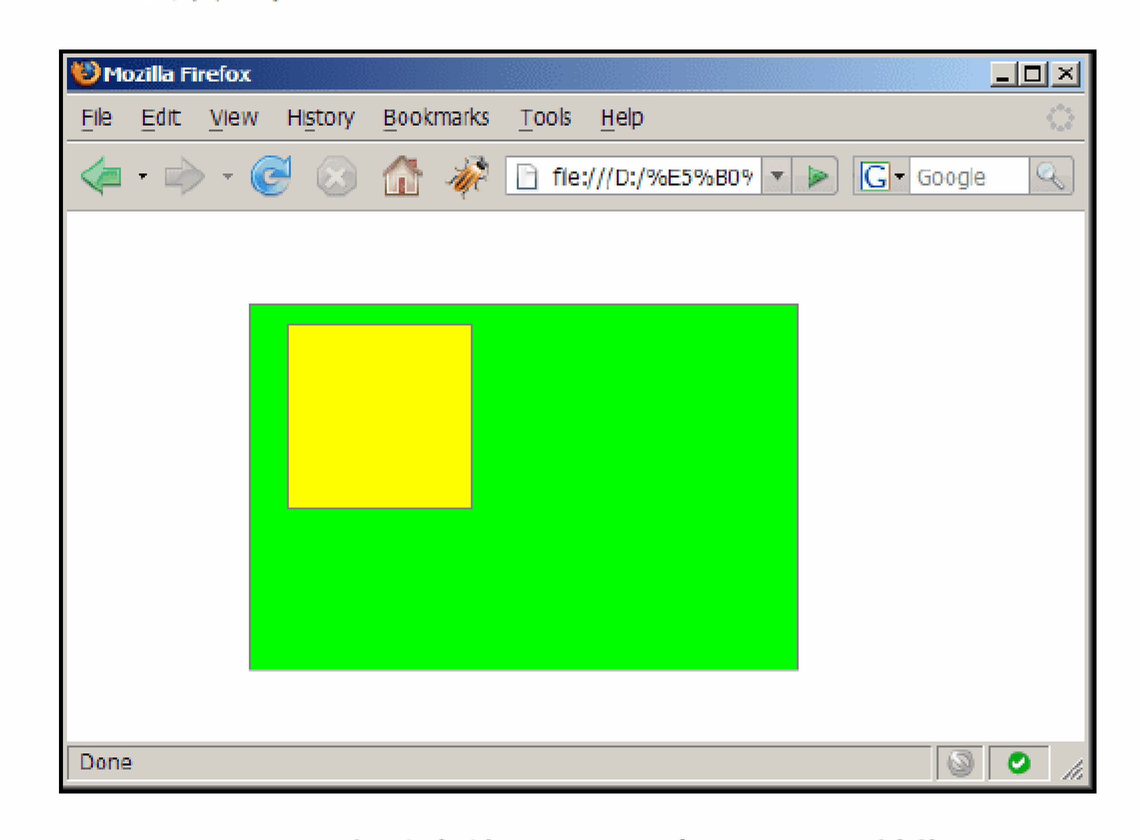


图 17-9　绝对定位 XML 元素(Firefox 浏览器)

5. 固定定位

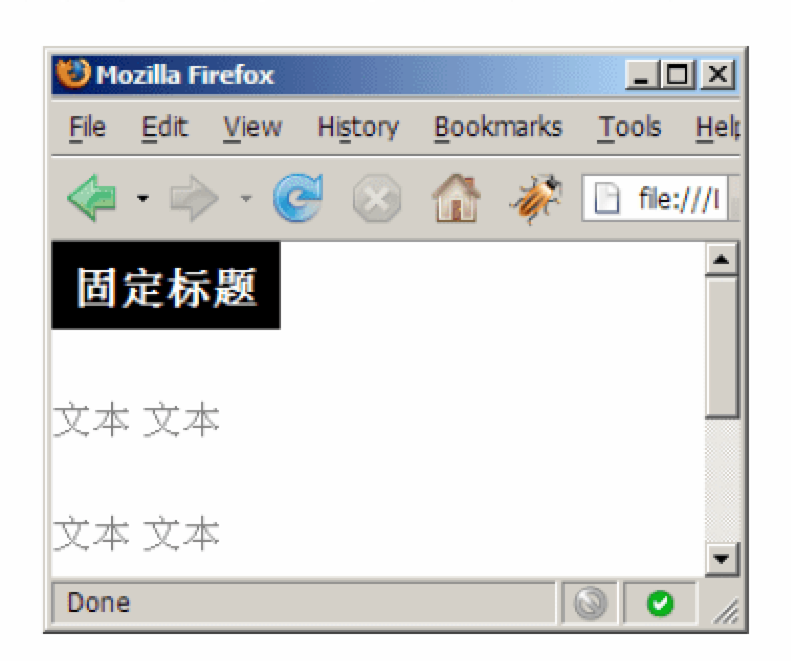
固定定位元素以浏览器窗口为基准进行定位，不受滚动条的影响。注意 IE 不支持固定定位。在下面的示例中，title 将被设为固定定位，保持在浏览器窗口的顶端。代码如下：

```
<document>
    <title>固定标题</title>
    <paragraph>文本 文本</paragraph>
    <paragraph>文本 文本</paragraph>
    <paragraph>文本 文本</paragraph>
    <paragraph>文本 文本</paragraph>
    <paragraph>文本 文本</paragraph>
</document>
```

样式代码如下：

```
title{
    position:fixed;
    height:20px;
    padding:10px;
    background:black;
    color:white;
    font-size:20px;
    font-weight:bold;
    top:0;
}
paragraph{
    display:block;
    margin:10px 0;
    padding:10px 0;
    font-size:18px;
    color:gray;
}
```

图 17-10 显示了 Firefox 浏览器对此文档的显示效果。固定定位元素以浏览器窗口为基准进行定位，即使滚动条发生移动，固定定位元素仍保持不动。



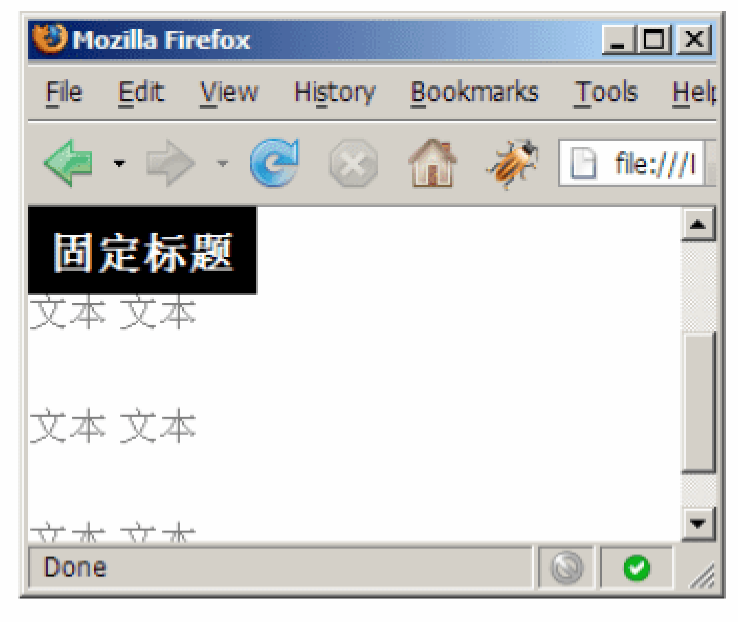


图 17-10 固定定位(Firefox 浏览器)



17.2　用于打印的 CSS

17.2.1　为打印媒介指定 CSS

除了通过浏览器查看 Web 页面外，有时候我们还需要将页面内容进行打印输出。比如一些自动化办公系统就向用户提供了打印数据的功能。CSS 除了可以控制页面如何在浏览器中显示，还可以控制打印输出的样式。打印输出的内容通常与浏览器的显示效果不同，比如页面的导航菜单，背景颜色和图像都不会被打印出来。CSS 可以把与打印无关的内容排除在外。

1. media 属性

打印机和显示属于不同的显示媒介(Media)，为了区分不同的媒介，(X)HTML 提供了 media 属性，该属性可用于 link 元素或者 style 元素。media 默认属性值为 all，表示样式将用于包含显示器、打印机在内的所有媒介中，属性值 print 则可指定专门用于打印输出的样式，比如：

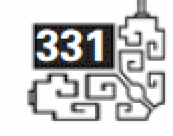
```
<link media="print" type="text/css" rel="stylesheet" href="print.css" />
```

或者：

```
<style media="print" type="text/css">
...
</style>
```

属性值取 screen 则表明样式将用于显示输出。通过 media 属性，我们就能为同一页面在不同的媒介上指定特定的样式。请看示例：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>style 元素的 media 属性</title>
<style type="text/css" media="screen">
p{
    padding:10px;
    width:300px;
    background:yellow;
    border:2px dotted gray;
    text-indent:2em;
    line-height:1.4em;
}
</style>
<style type="text/css" media="print">
p{
    text-indent:2em;
    font-size:11pt;
```



```
}
</style>
</head>

<body>
<p>使用(X)HTML的media属性就可以实现为不同的媒介指定特定的CSS样式。media默认属性值为all，表示样式会应用于所有的媒介，包括显示器、打印机等，属性值screen则表示样式只用显示器，print表示样式将会应用于打印输出。</p>
</body>
</html>
```

以上代码有两个 style 元素，每个 style 元素指定了不同的 media 属性值，因此第一个 style 元素中的样式只针对显示器，而第二个 style 元素只针对打印机。图 17-11 和图 17-12 分别为浏览器上的显示效果和打印预览中的打印效果。

使用(X)HTML的media属性就可以实现为不同的媒介指定特定的CSS样式。media默认属性值为all，表示样式会应用于所有的媒介，包括显示器、打印机等，属性值screen则表示样式只用显示器，print表示样式将会应用于打印输出。

图 17-11　用于显示器的 CSS 样式效果

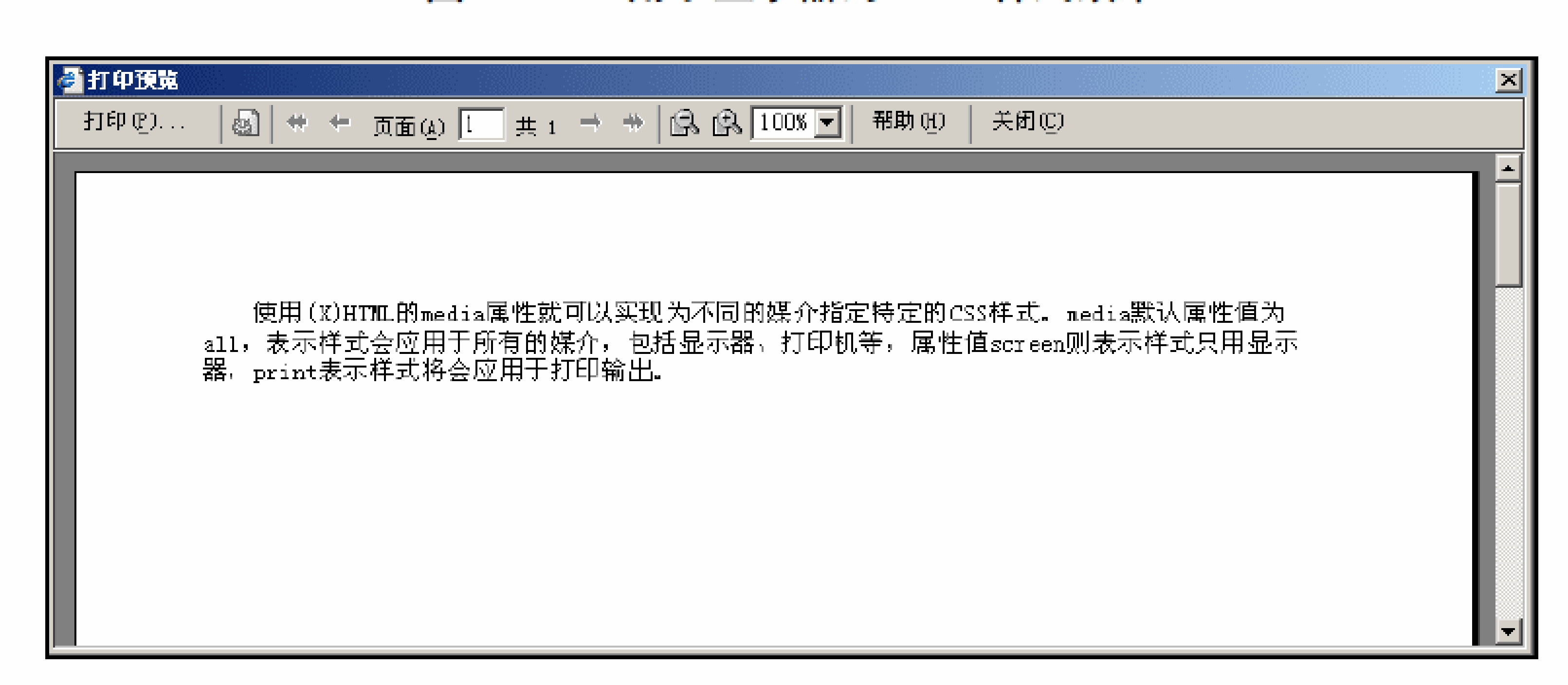


图 17-12　用于打印机的 CSS 样式效果(通过 IE 的打印预览功能查看)

2. @media 规则

除了使用 media 属性外，CSS 提供的@media 规则同样可以为样式指定不同的应用媒介。请看下面这个示例：

```
@media screen{
    p{
        width:300px;
        background:yellow;
        border:1px solid red;
        text-indent:2em;
        padding:10px;
```



```
    }
}
@media print{
    p{
        width:300px;
        border:0.02cm solid gray;
        text-indent:2em;
    }
}

<p>使用(X)HTML 的 media 属性就可以实现为不同的媒介指定特定的 CSS 样式。media 默认属性值为 all，表示样式会应用于所有的媒介，包括显示器、打印机等，属性值 screen 则表示样式只用显示器，print 表示样式将会应用于打印输出。除此之外，CSS 提供的@media 规则同样可以为样式指定不同的应用媒介。</p>
```

这次，针对不同的 CSS 样式分别放在不同的@media 规则下，图 17-13 和图 17-14 分别为显示器和打印预览的显示效果。

使用(X)HTML的media属性就可以实现为不同的媒介指定特定的CSS样式。media默认属性值为all，表示样式会应用于所有的媒介，包括显示器、打印机等，属性值screen则表示样式只用显示器，print表示样式将会应用于打印输出。除此之外，CSS提供的@media规则同样可以为样式指定不同的应用媒介。

图 17-13　@media 规则为 screen 的样式效果

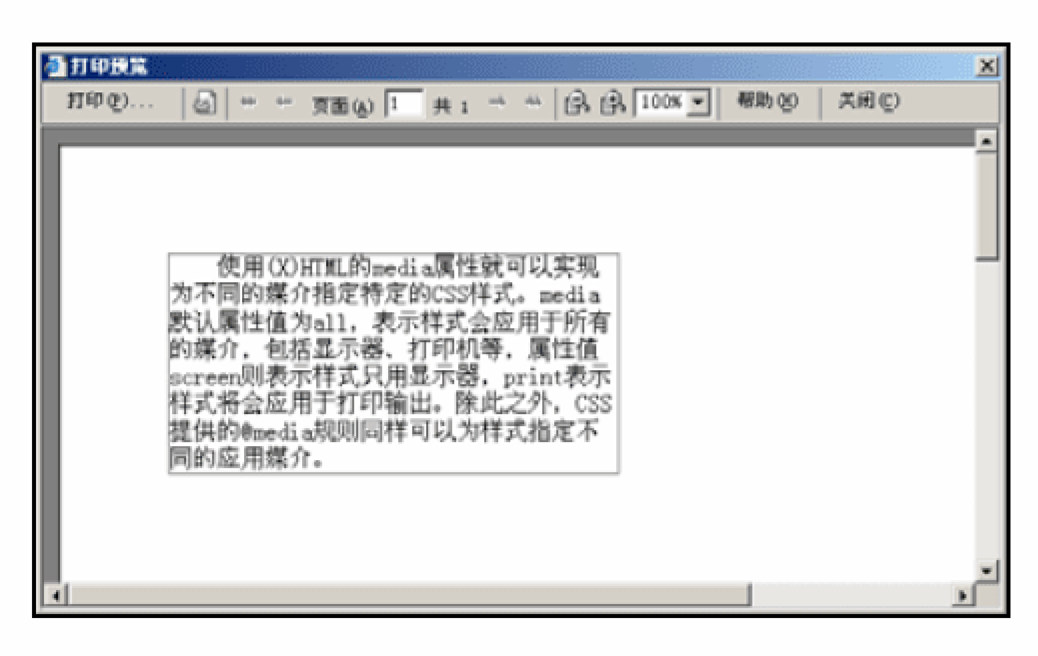


图 17-14　@media 规则为 print 时的样式效果

3. @import 规则的媒介限制

前面讲过，CSS 的@import 规则可以在 CSS 文件中导入其他外部的 CSS 样式文件，通过在该规则后添加媒介限制可以将不同的外部样式应用到不同的媒介中，比如：

```
@import url(screen.css) screen;
@import url(print.css) print;
```

注意： IE 浏览器对@import 规则支持不够完善，它无法导入带有媒介限制的外部样式文件。因此可利用这一点为非 IE 浏览器添加特定的样式。

17.2.2 分页处理

当打印一份文档时，其内容可能会占用多个页面，而浏览器的显示是连续的，没有页的概念。在打印文档时，我们希望能够控制如何分页，比如让标题独占一个页面。CSS 提供了 page-break-before 和 page-break-after 用来控制分页，它们允许的属性值为：auto、always、avoid、left 和 right，默认值为 auto。

auto 值表示元素前后不强制设置分页方式，打印机会根据内容多少自动控制分页。always 表示元素前后均设置分页。

在下面的示例中，页面提供了某产品的使用手册，当打印时希望使用手册的标题作为封面，其余内容要另起一页：

```
<h1 class="mainTitle">使用手册</h1>
<h2>一 基本功能的使用方法</h2>
<ul>
    <li>开机和关机</li>
    <li>设置的基本操作</li>
    <li>文件的基本操作</li>
    <li>如何充电</li>
</ul>
```

CSS 代码如下：

```
@media print{
    h1.mainTitle{
        text-align:center;
        font:60pt "黑体";
        letter-spacing:0.6em;
        page-break-after:always;
        margin-top:6cm;
    }
    h2{
        font-size:18pt;
    }
    li{
        font-size:16pt;
        line-height:1.5em;
    }
}
```

样式代码设定标题内容之后强制分页。通过打印预览可以看到如下效果(见图 17-15)。

属性值 left 和 right 表示元素需独自占用左侧或右侧页面。假如我们都想让使用手册中每一部分的标题打印在右侧的页面，则可添加如下样式：

```
h2 {
    page-break-before:left;
    page-break-after:always;
}
```



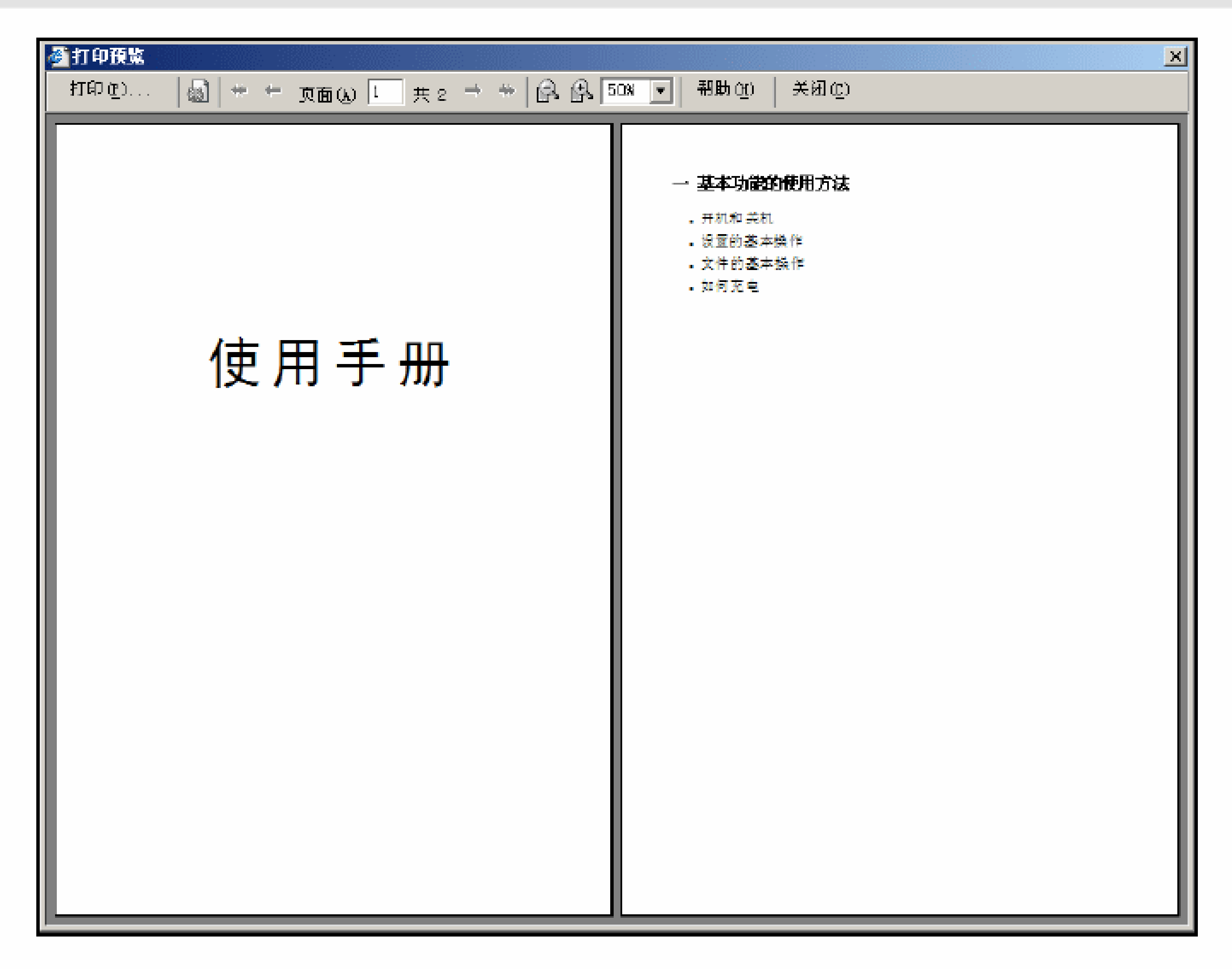


图 17-15　使用手册四个字后面的内容被放到下一页

这样，当 h2 元素之前的元素位于左侧页面时，h2 的内容将会从右侧页面开始显示；如果 h2 之前的元素也位于右侧页面，那么 h2 元素将会显示在下一个右侧的页面上，中间会空出一个左侧页面。

属性值 avoid 正好和 always 相反，它要求该元素与前后元素之间不产生分页。但是浏览器对此也只能尽力而为，假如元素后面跟随着一张很大图像，则浏览器不得不将此图像放置在下一个页面中显示。

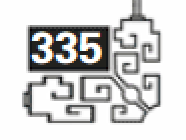
除了 page-break-before 和 page-break-after 属性外，CSS 还提供了一个 page-break-inside 属性，它的属性值只有 auto 和 avoid，默认值为 auto。avoid 表示尽量不要在元素之间产生分页，同样，浏览器只是尽最大努力保证不在元素之间产生分页。

CSS 用于控制分页的属性只是针对具有分页效果的媒介，对显示器的显示效果不会产生任何影响。

17.3　用户界面元素

17.3.1　鼠标指针

当使用桌面应用程序时，不同鼠标指针的样式通常代表着不同的软件行为，比如在输入框中的鼠标指针和调整窗口大小时的鼠标指针就具有不同的样式。假如读者使用过 Photoshop 之类的图像处理软件，会发现程序提供了丰富的鼠标指针样式(比如笔刷、橡皮擦等)来表示当前的用户操作。然而，在开发 Web 应用程序时，鼠标指针却经常被忽略。如果能根据使用环境来改变鼠标指针的样式，对于增强 Web 页面的交互性、改善用户体验，都是大有帮助的。



1. cursor 属性

CSS 中的 cursor 属性可以控制页面中鼠标的指针样式。表 17-2 总结了规范中定义的鼠标指针以及它们的含义。

表 17-2　cursor 属性值的含义

属 性 值	含 义
auto	浏览器决定鼠标指针样式
crosshair	十字形鼠标指针
default	默认鼠标指针样式
pointer	表示链接的鼠标指针
move	表示移动的鼠标指针
e-resize、ne-resize、nw-resize、n-resize、se-resize、sw-resize、s-resize、w-resize	表示某一方向的边将要移动，前缀表示了边的方向，比如 e 表示 east，即右侧。sw 表示 south west 为左下方
text	表示可选择文本，通常为竖型指针
wait	表示程序忙，需要等待，通常为沙漏或表
help	当前对象有相应的帮助信息，通常为问号
progress	表示程序后台正在进行处理，通常为默认指针和沙漏的组合

图 17-16 展示了 Windows 系统中不同鼠标指针的样式。

crosshair
default
pointer
move
e-resize w-resize
ne-resize sw-resize
nw-resize se-resize
n-resize s-resize
text
wait
help
progress

图 17-16　cursor 属性值对应的鼠标指针样式(Windows 系统)

当给元素添加鼠标指针样式后，只要访问者将鼠标移到该元素上，鼠标指针就会按照 cursor 属性设定的样式进行显示。请看示例：

```
div{
    font-size:12px;
```

```
    font-family:"Times New Roman", "宋体", serif;
}
div.section{
    width:200px;
    border:1px solid #035F30;
}
div.header{
    cursor:move;
    padding:4px;
    font-weight:bold;
    background:#EAF6F0;
    border-bottom:1px solid #035F30;
}
div.body{
    padding:4px;
    line-height:1.4em;
    background:#FAFAFA;
}

<div class="section">
    <div class="header">天气预报</div>
    <div class="body">
        晴<br />
        最高气温：25 度<br />
        最低气温：13 度<br />
        风力：1-2 级
    </div>
</div>
```

以上代码设计了一个可移动的天气预报板块，当用户鼠标移至标题上时，按下鼠标可以移动整个板块(可通过 JavaScript 实现)。为了让用户体验到移动功能的存在，需要将标题区域的鼠标指针样式设为 move，表示拖动该区域会产生移动效果。图 17-17 显示了代码的效果。

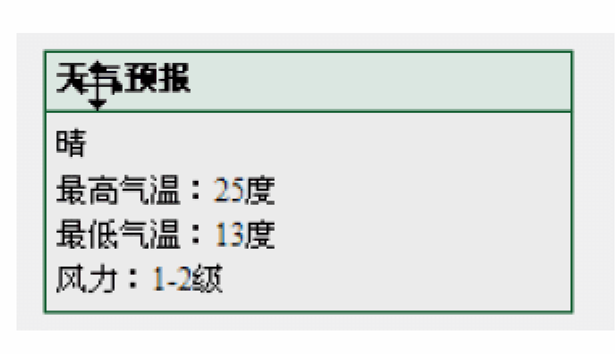
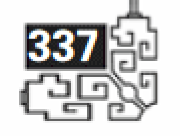


图 17-17　将鼠标指针样式设为 move

图 17-17 取自 IE 浏览器，我们发现只有当鼠标指针位于文字上方时，指针样式才会发生改变，否则还是默认样式。而在 Firefox 浏览器中，只要鼠标位于该 div 元素区域中，指针都会成为移动的样式。为了解决 IE 存在的问题，需要给该 div 元素设定确切的宽度值，于是添加如下声明：

```
width:192px;
```

宽度设为 192px，再加上左右填充各有 4px，一共 200px，和父元素的宽度一致。现在的效



果如图 17-18 所示，鼠标指针在 div 内任何区域都能改变样式了。

天气预报

晴
最高气温：25度
最低气温：13度
风力：1-2级

图 17-18　添加确定的宽度之后，鼠标指针能在 div 元素任何区域内显示为移动样式

2. 非标准的扩展 cursor 属性值

前面介绍的鼠标指针均是 CSS 规范中定义的，除此之外，还有一些非标准的属性值，它们绝大部分可用于 Windows 平台下的 Firefox 和 IE 浏览器中。表 17-3 包含了这些非标准的属性值及其含义。

表 17-3　cursor 扩展属性值含义

属 性 值	含　义
hand	手型指针
all-scroll	表示全方向可滚动的指针
col-resize	调整列宽度时的指针
row-resize	调整行高度时的指针
no-drop	表示不可访问的指针
not-allowed	表示不允许的指针
vertical-text	表示竖型文本的指针

图 17-19 展示了以上扩展属性值在 Windows 平台下对应的鼠标指针样式。

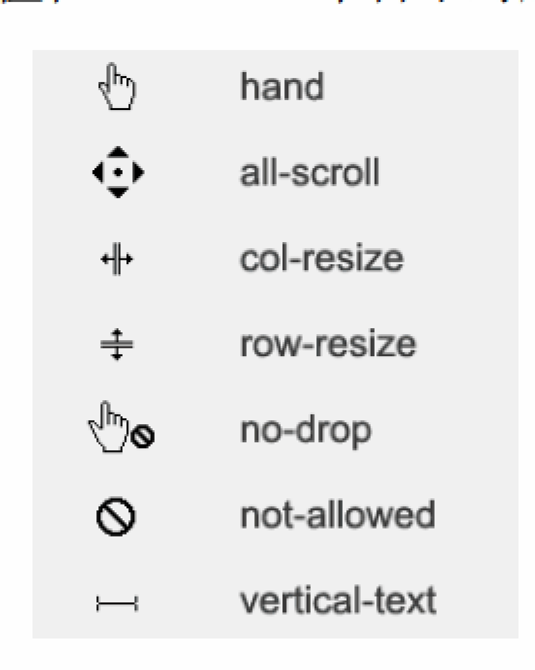


图 17-19　扩展鼠标指针样式(Windows 系统)

如果你的网页只针对 Window 平台下使用 IE 和 Firefox 浏览器的用户，就可以大胆地使用这些非标准的扩展属性值。

3. 自定义指针样式

除了指针样式关键字外，cursor 属性值还可以是一个 URL，可使用该 URL 所指向的图像作为鼠标指针。Firefox 支持 PNG、GIF 和 JPEG 等格式的图像作为鼠标指针，而 IE 则只支持



Windows 平台下专用的鼠标指针文件(后缀为 cur)。不是所有浏览器都支持自定义鼠标，为了兼容这些浏览器，需要在 URL 后再添加一个标准的指针关键字。请看示例：

```
div{
    cursor:url(images/arrow.gif), default;
    width:200px;
    height:100px;
    border:1px solid gray;
}

<div></div>
```

图 17-20 为 Firefox 浏览器中的效果，在 div 元素区域中，鼠标指针将变为自定义的图像。不支持自定义鼠标指针样式的浏览器则会按照 default 值显示默认的样式。



图 17-20　Firefox 浏览器中使用 GIF 格式图像作为鼠标指针样式

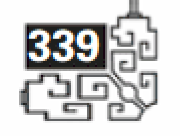
4. Mozilla 扩展中的鼠标指针样式

Mozilla 扩展提供了一系列的样式规则，它们只适用于 Firefox 浏览器，并不属于 CSS 标准规范(实际上一部分已经被提交至 W3C，推荐作为 CSS3 的标准属性，但是采纳的属性名称会有变化。后面将会详细介绍 Mozilla 的 CSS 扩展)。Mozilla 扩展中的名称均以“-moz”开头。表 17-4 总结了 Mozilla 扩展中提供的指针样式名及其含义。

表 17-4　Mozilla 扩展提供的鼠标指针样式属性值和含义

属 性 值	含　义
-moz-alias	表示将要创建一个对象的别名或快捷方式
-moz-cell	表示将要选择一个或一组单元格
-moz-context-menu	表示指针所指的对象包含右键菜单
-moz-copy	表示将要复制一个对象
-moz-grab	表示将要抓取一个对象
-moz-grabbing	表示正在抓取一个对象
-moz-spinning	表示浏览器正在后台进行处理
-moz-zoom-in	表示将要放大某对象
-moz-zoom-out	表示将要缩小某对象

图 17-21 展示了 Windows 平台下各个关键字所对应的指针样式。



-moz-alias
-moz-cell
-moz-context-menu
-moz-copy
-moz-grab
-moz-grabbing
-moz-spinning
-moz-zoom-in
-moz-zoom-out

图 17-21　Mozilla 扩展提供的鼠标指针样式(Windows 系统)

17.3.2　系统颜色

除了可以使用颜色关键字和 RGB 值改变元素颜色外，CSS 还提供了若干系统颜色关键字，可使页面的色彩效果能与访问者的系统的完全一致。如果某系统中没有关键字所对应的颜色，则该颜色将被赋予一个与其最为接近的颜色值或默认值。表 17-5 总结了这些关键字以及它们所代表的含义。注意，关键字是不区分大小写的，但是将每个单词首字母大写有助于提高代码的可阅读性。

表 17-5　系统颜色关键字

关 键 字	含　义
ActiveBorder	活动窗口的边框
ActiveCaption	活动窗口的标题
AppWorkspace	多文档界面中的背景颜色
Background	桌面的背景颜色
ButtonFace	三维元素的正面的颜色
ButtonHighlight	三维元素的高光部分的颜色(面向光源的边的颜色)
ButtonShadow	三维元素的阴影部分的颜色
ButtonText	按钮上文本的颜色
CaptionText	标题中文本的颜色
GrayText	灰度颜色(表示禁用)，
Highlight	控件中选中项的颜色
HighlightText	控件中选中项的文本颜色
InactiveBorder	非活动状态窗口的颜色
InactiveCaption	非活动状态窗口的标题颜色
InactiveCaptionText	非活动状态窗口的标题文本颜色
InfoBackground	工具提示(tooltip)的背景颜色

续表

关 键 字	含　义
InfoText	工具提示的文本颜色
Menu	菜单背景色
MenuText	菜单中文本的颜色
Scrollbar	滚动条中灰色区域的颜色
ThreeDDarkShadow	三维元素的深色阴影部分的颜色
ThreeDFace	三维元素的正面的颜色
ThreeDHighlight	三维元素的高光部分的颜色
ThreeDLightShadow	三维元素的向光部分的颜色(面向光源的边的颜色)
ThreeDShadow	三维元素的阴影部分的颜色
Window	窗口背景的颜色
WindowFrame	窗口边框的颜色
WindowText	窗口文本的颜色

请看下面这个示例：

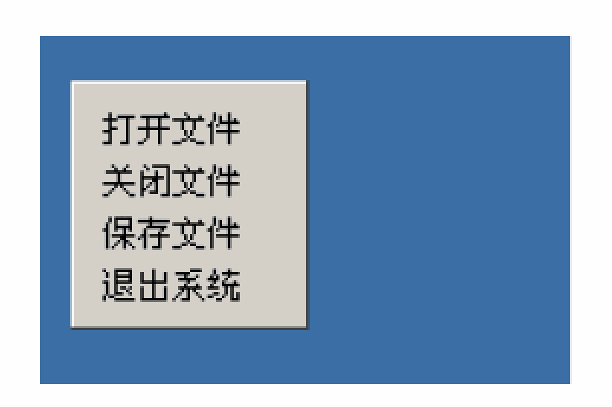
```
body{
    background:Background;
}
ul{
    background-color:Menu;
    width:60px;
    padding:6px 10px;
    margin:0;
    border-top:1px solid ThreeDHighlight;
    border-left:1px solid ThreeDHighlight;
    border-right:1px solid ThreeDDarkShadow;
    border-bottom:1px solid ThreeDDarkShadow;
}
ul li{
    color:MenuText;
    font:menu;
    list-style:none;
}

<ul>
    <li>打开文件</li>
   <li>关闭文件</li>
   <li>保存文件</li>
   <li>退出系统</li>
</ul>
```

以上代码分别给页面背景色、ul 元素背景色和 li 元素文字颜色指定为系统的桌面颜色、菜单背景色和菜单文本的颜色。ul 元素的左侧和上侧边框颜色使用高光颜色，右侧和下侧边框使用了深色阴影的颜色。li 元素中 font 属性为 menu，还记得这是什么含义吗？在第 9 章我们讲



过，font 属性也可以使用若干表示系统字体属性的关键字，menu 的含义是使用系统菜单文本的字体属性。图 17-22 为代码的效果(该效果在读者的计算机上可能会有差异，因为系统颜色是可以自行设置的)。



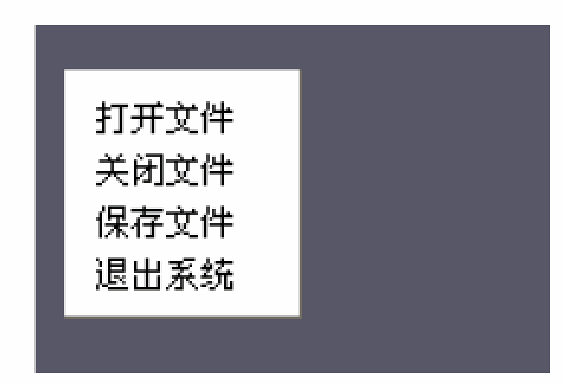


图 17-22 使用系统颜色(左图为使用 Windows 经典主题，右图使用了 Windows XP 的银色主题)

17.3.3 轮廓线

有时，Web 设计者希望为按钮、活动的表单元素添加一个轮廓线(Outline)，用来突出显示它们。轮廓线也是经常被忽略的界面元素，适当地使用它可以提高应用程序的易用性，增强用户体验。轮廓线与边框的不同之处在于以下两个方面：

- 轮廓线不占用任何空间(不影响元素的大小)。
- 轮廓线可以不是矩形的。

CSS 提供了 outline-color、outline-style、outline-width 以及一个简写的 outline 属性来控制元素的轮廓线样式。

outline-color 属性可接受所有表示颜色的属性值，此外还可使用 invert 关键字，它表示将轮廓线区域下面的颜色进行反转。outline-style 可使用除 hidden 外所有 border-style 所使用的属性值。outline-width 可用的属性值与 border-width 属性的属性值完全一致。outline 属性为简写形式，可同时包含三个样式属性。

注意：四个方向的轮廓线样式是一致的，这一点与边框不同，CSS 中没有“outline-left”、“outline-top”一类的属性。

现在来看一个示例：

```
input{
    width:120px;
    height:14px;
    padding:3px 4px 4px 3px;
    border:1px solid gray;
    margin:3px;
}
input.current{
    outline:1px solid invert;
    background:#EDEDED;
}

<input class="current" type="text" tabindex="1" /><br />
<input type="text" tabindex="2" />
```



图 17-23 为 Firefox 浏览器中的效果(IE 浏览器不支持轮廓线属性)。invert 属性值将原来的白色反转为黑色。

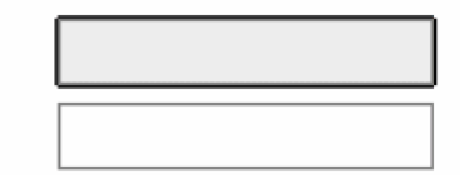


图 17-23　outline 属性的使用

假如我们的页面背景颜色指定为黑色，则轮廓线自动反转为白色(见图 17-24)。



图 17-24　invert 属性值会自动反转原始颜色

17.4　滤镜与转场

滤镜(Filter)与转场(Transition)是 IE 浏览器所特有的，不属于 CSS 的标准规范，但是 IE 浏览器有着相当高的市场占有率，使用滤镜和转场能快速有效地实现某些页面特效。

本节将介绍滤镜和转场的基本概念和部分属性的使用方法，有关滤镜的完整信息可参考微软的 msdn 网站：

```
http://msdn2.microsoft.com/en-us/library/ms532853(VS.85).aspx
```

滤镜和转场都是通过 CSS 的 filter 属性来赋值的，注意这个属性不是 CSS 规范中的标准属性，只是作为 IE 浏览器的扩展存在。filter 属性中包含滤镜的描述和该滤镜所能使用的属性。添加滤镜的语法规则如下：

```
filter: progid:DXImageTransform.Microsoft.filtername(sProperties)
```

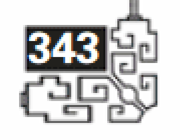
17.4.1　程序生成面

所谓程序生成面(Procedural Surface)是指存在于元素内容与元素背景之间一个假想的界面。在界面区域内，每个像素的颜色值和透明度都可以动态改变。本小节介绍的两个滤镜都作用于程序生成面上，并不影响元素本身。

1. AlphaImageLoader

AlphaImageLoader 滤镜可以在元素内容与背景之间添加图像，同时对图像进行剪裁和改变大小操作。这个滤镜也可用来解决 PNG 格式图像无法在 IE 中实现透明效果的问题。

在下面这个示例中，我们通过此滤镜添加一幅图像：



```
p{
    width:200px;
    background:#99FF99;
    padding:10px;
    color:red;
    filter:progid:DXImageTransform.Microsoft.AlphaImageLoader(
          src="images/scene.jpg", sizingMethod='scale');
}

<p>使用 AlphaImageLoader 滤镜可以给元素动态添加图像。图像被加在元素内容和元素背景之间。
</p>
```

效果如图 17-25 所示，src 属性指定的图像被添加在元素背景之上，sizingMethod 属性指定了图像要根据元素大小进行缩放。



图 17-25　使用 AlphaImageLoader 滤镜添加图像

接下来，我们通过该滤镜为元素增加 PNG 格式的图像，这时 PNG 图像的透明信息能正常显示，从而可作为 IE 下 PNG 图像的无法透明问题的解决方案：

```
div{
    float:left;
    width:200px;
    height:160px;
    margin:10px;
    background:green;
}
div#pngHackForIE6{
    filter: progid:DXImageTransform.Microsoft.AlphaImageLoader(
            enabled="true", src="images/test transparent.png");
}

<div id="pngInIE6"><img src="images/test_transparent.png" /></div>
<div id="pngHackForIE6"></div>
```

以上代码效果如图 17-26 所示，左侧的图像是作为 img 元素插入的，PNG 透明区域显示为灰色，右侧图像是通过 AlphaImageLoader 滤镜添加的，具有透明效果。

注意：使用 AlphaImageLoader 滤镜的元素中如果包含链接和表单元素，则它们会失去其原有的功能，不能响应用户的任何操作。解决办法是在使用该滤镜的元素中再添加一个新元素作为容器，并给新元素添加声明 position:relative，则链接和表单的功能可恢复正常。怪不得有人说 IE 简直是糟糕透了。



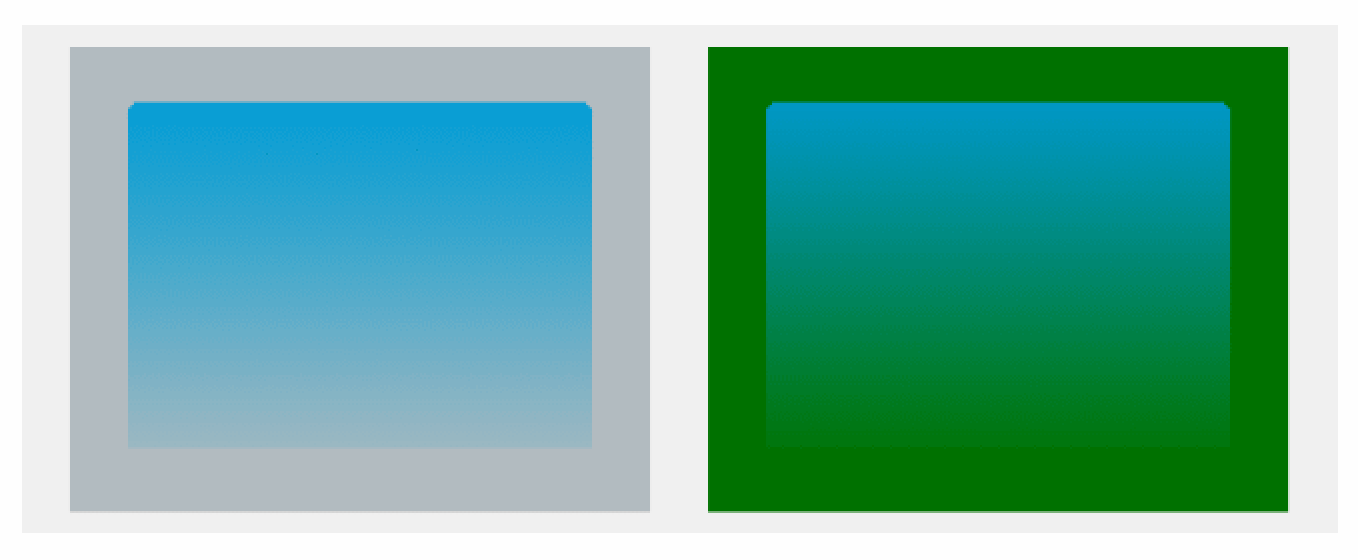


图 17-26　使用 AlphaImageLoader 滤镜解决 IE6 无法正确显示 PNG 透明效果的问题

2. Gradient

Gradient 滤镜可在元素背景与内容之间添加颜色，确定颜色的透明度以及是否有渐变效果。属性 startColorstr 与 endColorstr 用来设定颜色渐变的开始色彩和结束色彩。例如：

```
p{
    padding:6px 0 4px 5px;
    height:100%;    /* needed for filter */
    font-size:12px;
    border:1px solid #999999;
    filter:progid:DXImageTransform.Microsoft.gradient(
            startColorstr="#11000000", endColorstr="#44000000",
        GradientType="0");
}

<p>使用 Gradient 滤镜可以给元素动态添加颜色。颜色可拥有透明度和渐变效果。</p>
```

注意到表示颜色的字符串格式为#AARRGGBB，后面的 RRGGBB 和 CSS 中的颜色格式一致，分别代表红、绿、蓝三种颜色值，前面的 AA 表示透明程度，00 为完全透明，FF 为完全不透明。代码的效果如图 17-27 所示。

使用Gradient滤镜可以给元素动态添加颜色。颜色可拥有透明度和渐变效果。

图 17-27　使用 Gradient 滤镜实现渐变效果

17.4.2　静态滤镜

所有的滤镜和转场本质上都是滤镜，都是通过 filter 属性将效果应用到元素上的。本节介绍的静态滤镜是指滤镜所产生的效果是静止不动的。

1. Alpha

Alpha 滤镜可以将元素内容设置为透明，opacity 属性决定了元素的透明程度。例如：

```
div#background{
    background:url(images/backwall.jpg);
    width:200px;
    height:120px;
```



```
}
div#inner{
    width:80px;
    padding:8px;
    background:yellow;
    filter:progid:DXImageTransform.Microsoft.Alpha(opacity="40");
}

<div id="background"><div id="inner">此元素内容具有半透明效果。</div></div>
```

效果如图 17-28 所示。



图 17-28　Alpha 滤镜可使元素内容呈现半透明效果

2. Blur

Blur 滤镜能使元素产生模糊效果，pixelradius 属性可设定模糊的程度。例如：

```
img#blur{
    filter:progid:DXImageTransform.Microsoft.Blur(pixelradius="6");
}

<img src="images/scene.jpg" />
<img id="blur" src="images/scene.jpg" />
```

图 17-29 中右侧的图像使用了 Blur 滤镜。




图 17-29　Blur 滤镜可使元素产生模糊效果

3. DropShadow

DropShadow 滤镜可给元素添加投影效果，投影的颜色、方向、偏移量都可以调节。例如：

```
img{
    filter:progid:DXImageTransform.Microsoft.DropShadow(
            offX="6", offY="6", color="gray", positive="true");
}

<img src="images/scene.jpg" />
```

图 17-30 为阴影效果。



图 17-30　DropShadow 滤镜为元素添加投影

4. Emboss 和 Engrave

Emboss 和 Engrave 滤镜分别产生浮雕和雕刻的效果，图像将以灰度方式显示。

5. Glow

Glow 滤镜会在元素周围增加发光效果，发光颜色和发光区域的大小都可以设置。例如：

```
body{
    background:gray;
}
p{
    width:100px;
    height:40px;
    color:white;
    background:orange;
    text-align:center;
    padding-top:20px;
    filter:progid:DXImageTransform.Microsoft.Glow(
            color="yellow", strength="10");
}

<p>发光效果</p>
```

在如图 17-31 所示的效果中，黄色的部分就是滤镜产生的发光效果。



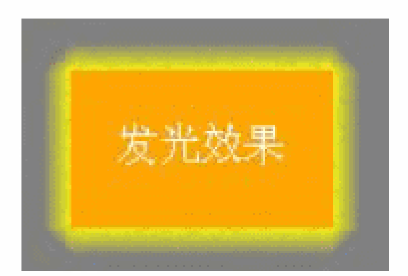


图 17-31　Glow 滤镜产生的发光效果

6. Shadow

Shadow 滤镜和 DropShadow 滤镜的效果差不多，但它产生的阴影会有渐变的效果，请看示例：

```
img{
    filter:progid:DXImageTransform.Microsoft.Shadow(
            direction="135" color="gray", strength="10");
}

<img src="images/scene.jpg" />
```

效果如图 17-32 所示。

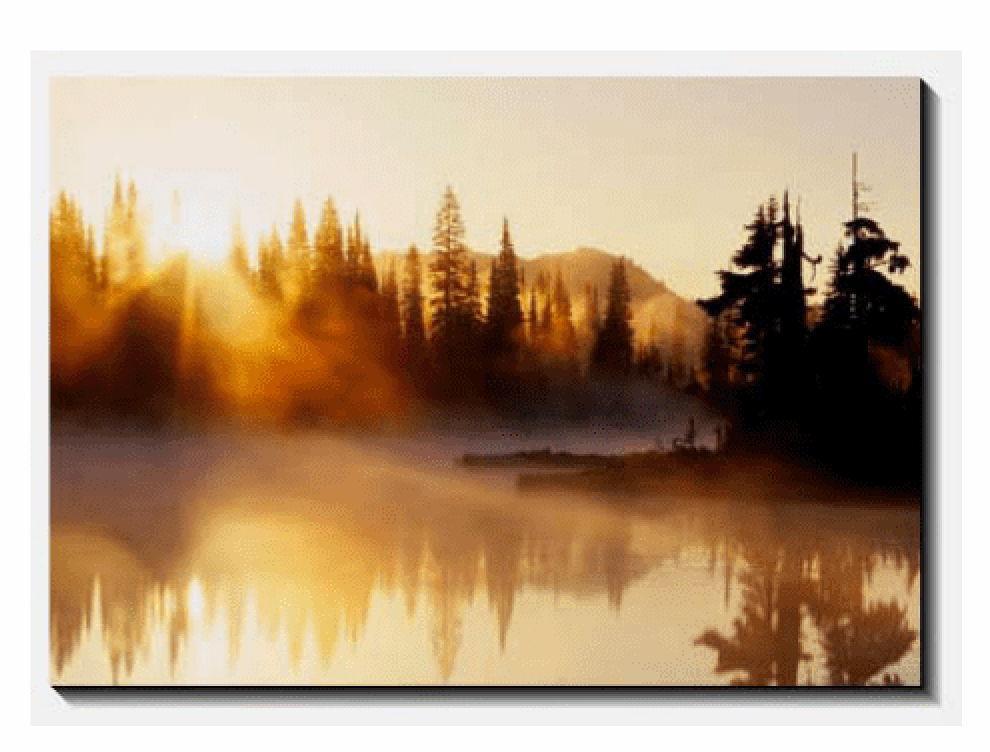


图 17-32　Shadow 滤镜产生的阴影效果

17.4.3　转场

转场实际上也是滤镜，它一般结合脚本(比如 JavaScript)一起使用，可产生动态变化的效果。

1. Barn

Barn 转场会产生一种开关门的效果。请看示例：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>Untitled Document</title>
<script language="javascript" type="text/javascript">
```



```
var bToggle = 0;
function fnToggle() {
    oDiv.filters[0].Apply();
    if (bToggle) {
        bToggle = 0;
        oDiv.style.backgroundColor="gold";}
    else {
        bToggle = 1;
        oDiv.style.backgroundColor="blue";}
    oDiv.filters[0].Play();
}
</script>
<style type="text/css">
div#oDiv{
     height:200px;
     width:200px;
     background-color: gold;
     filter:progid:DXImageTransform.Microsoft.Barn(
             duration="1", motion="out");
}
</style>
</head>

<body>
<button onclick="fnToggle()">Barn 转场</button><br /><br />
<div id="oDiv"></div>
</body>
</html>
```

我们给出了 XHTML 的完整代码，其中包含一段 JavaScript 代码，代码中执行了滤镜的 Apply 和 Play 方法，从而可产生动态变化的效果。单击按钮会产生动画效果(见图 17-33)。

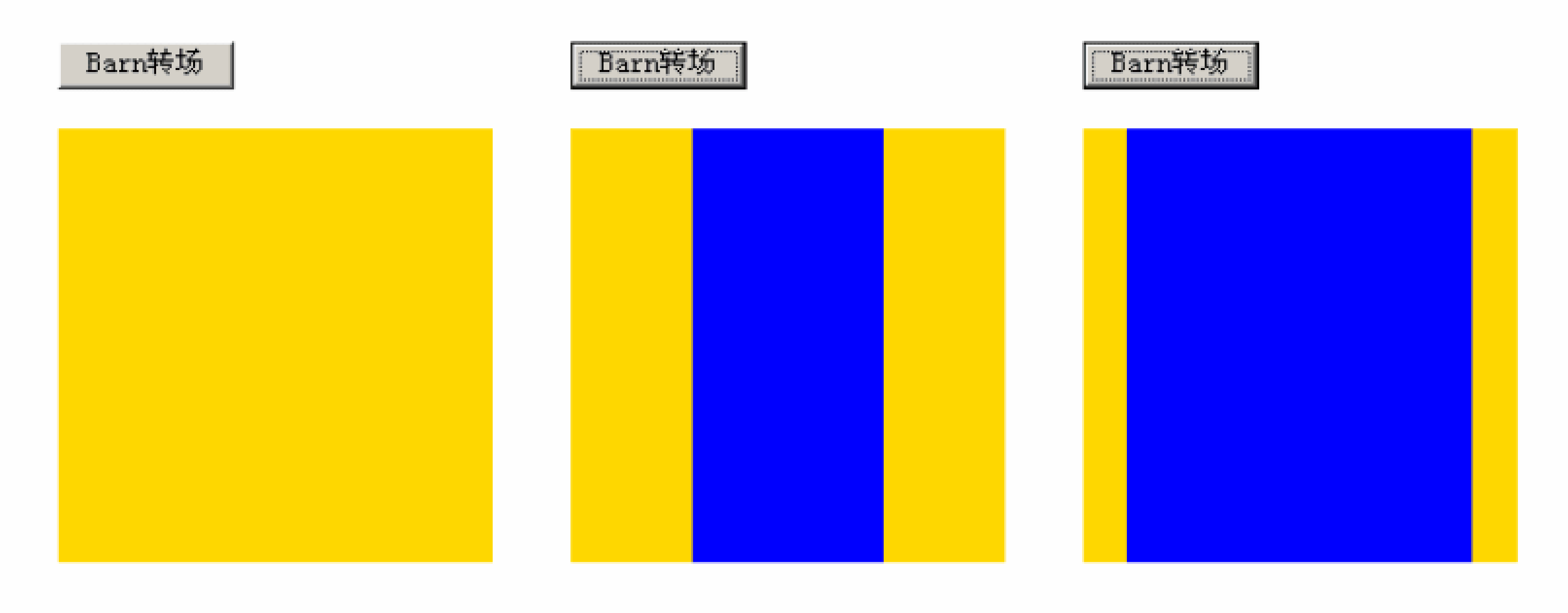


图 17-33　Barn 转场效果

其余转场的使用方式与此类似，不再举例说明。最后我们介绍一个 Pixelate 滤镜，它可以作为静态滤镜单独使用。



2. Pixelate

Pixelate 滤镜可将元素进行像素化处理，请看下面的示例：

```
img#image1{
    filter:progid:DXImageTransform.Microsoft.Pixelate(maxSquare="4");
}
img#image2{
    filter:progid:DXImageTransform.Microsoft.Pixelate(maxSquare="16");
}
img#image3{
    filter:progid:DXImageTransform.Microsoft.Pixelate(maxSquare="38");
}

<img id="image1" src="images/scene.jpg" />
<img id="image2" src="images/scene.jpg" />
<img id="image3" src="images/scene.jpg" />
```

三幅图像分别赋予了不同的像素化程度，效果如图 17-34 所示。



图 17-34　Pixelate 产生不同的像素化效果

17.5　behavior 属性与 CSS 表达式

17.5.1　behavior 属性

behavior 属性可以为元素添加行为，它的属性值是一个 URL，指向一个表示行为的资源或者行为名。有人通过 behavior 属性解决了 PNG 图像在 IE 中的透明显示问题。具体做法是：编写一个 HTC(HTML Component，HTML 组件)文件，在其中添加 JavaScript 脚本，给页面所有 img 元素、任何元素的背景图以及 CSS 添加的背景图设置 AlphaImageLoader 滤镜，然后再给特定的元素添加 behavior 属性，将这个组件导入到页面中。比如：

```
img, div{behavior: url(iepngfix.htc)}
```

当然也可以给所有元素增加该属性：

```
* {behavior: url(iepngfix.htc)}
```

HTML 组件是实现 DHTML 行为的一种方式，通过 behavior 属性就可以将行为引入到页面中。



延伸： 有关使用 behavior 属性和 HTML 组件解决 IE 的 PNG 问题的内容，请参考：
http://www.twinhelix.com/css/iepngfix
有关 HTML 组件的内容请参考：
http://www.w3.org/TR/NOTE-HTMLComponents

17.5.2　CSS 表达式

CSS 表达式(CSS Expression)是从 IE5 开始引入的，允许在 CSS 属性值中使用 JavaScript 表达式，它只适用于 IE 浏览器。属性值由表达式和其后括号中的脚本代码组成，脚本代码的结果会返回给该属性。在下面这个示例中，div 元素的宽度由表达式的计算结果来确定，它的宽度总是浏览器窗口宽度的一半：

```
div{
    height:80px;
    background:yellow;
    width:expression(document.body.clientWidth / 2);
}

<div>通过 CSS 表达式设置本元素宽度为浏览器窗口宽度的一半。</div>
```

图 17-35 为代码的效果。

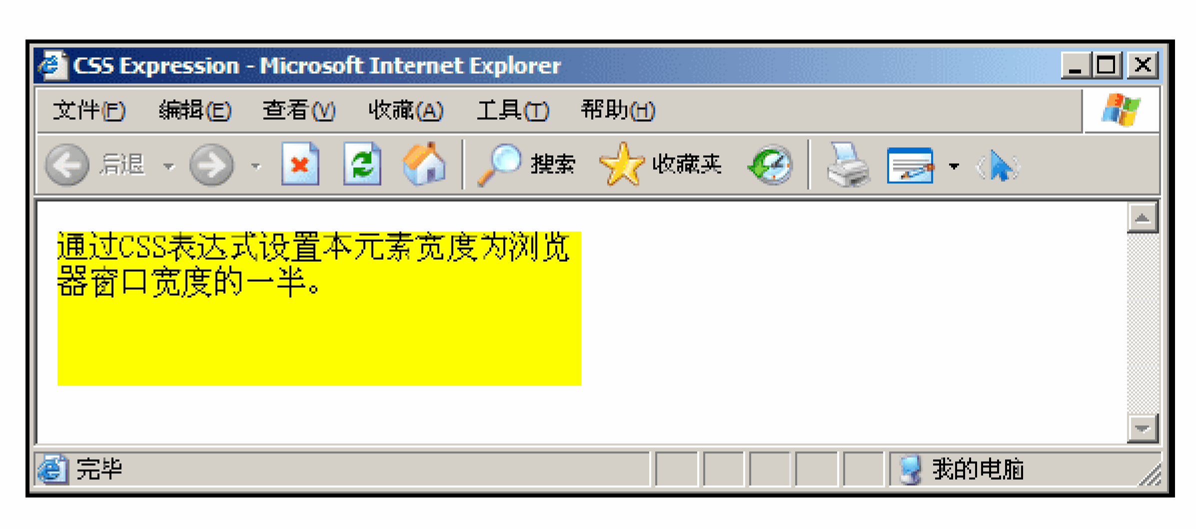


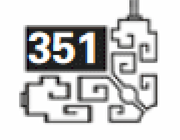
图 17-35　使用 CSS 表达式设置元素宽度为浏览器窗口宽度的一半

出于对性能的考虑，一些开发者不推荐使用 CSS 表达式。因为表达式的执行频率远比想象中的高，不但当页面加载时表达式需要执行，而且当窗口大小发生变化时，浏览器滚动条产生移动时，或是当用户将鼠标指针移到页面上时，表达式都需要重新计算。解决办法是改用 JavaScript 脚本对元素样式进行控制，以避免由 CSS 表达式带来的性能问题。

17.6　微软对 CSS 的扩展

17.6.1　控制滚动条外观

IE 5.5 及其后续版本支持通过 CSS 改变浏览器和 textarea 元素的滚动条的外观，这些 CSS 属性包括 scrollbar-3dlight-color、scrollbar-arrow-color、scrollbar-base-color、scrollbar-darkshadow-



color、scrollbar-face-color、scrollbar-highlight-color、scrollbar-shadow-color 以及 scrollbar-track-color。这些属性的属性值可以是任何有效的 CSS 颜色值。

需要注意的是，滚动条的外观属性只有在页面不包含任何文档类型声明时才会起作用，因此目前很少有人使用它。对于那些使用 Flash 技术制作的页面，文档类型声明的作用不是很重要。为了实现页面的一致性、增强美观程度，还是可以使用滚动条外观属性的。如图 17-36 所示为 Vabien Decoree(http://www.idecoree.com)公司的页面，其中滚动条的样式与页面整体风格搭配和谐，实现了外观的统一。

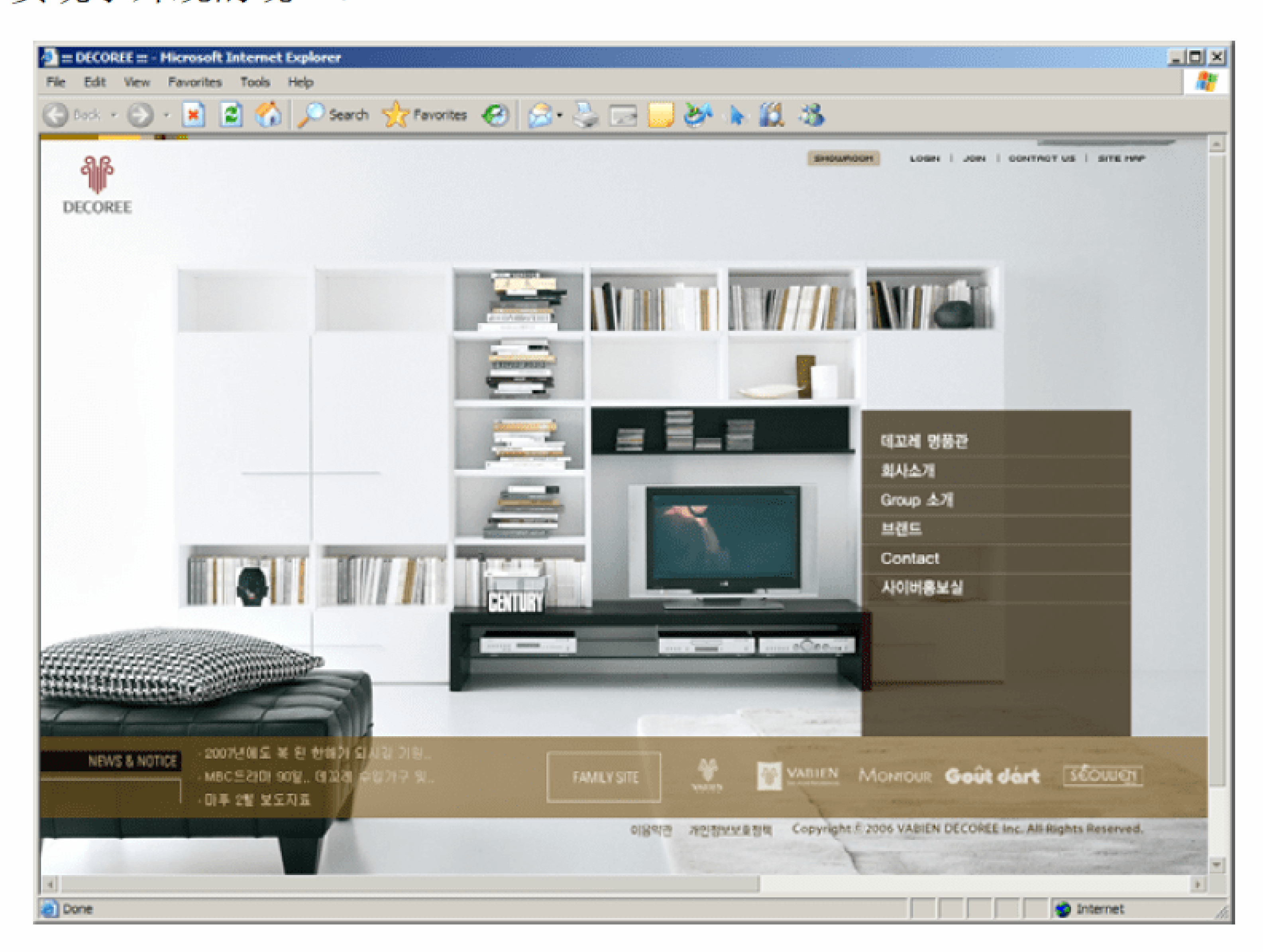


图 17-36 Vabien Decoree 的网站使用滚动条外观属性

下面是相关的 CSS 代码：

```
body{
    scrollbar-face-color:#d0d0c9;
    scrollbar-highlight-color:#cccccc;
    scrollbar-shadow-color:#8a8d7a;
    scrollbar-3dlight-color:#ffffff;
    scrollbar-arrow-color:#8a8d7a;
    scrollbar-track-color:#ecede7;
    scrollbar-darkshadow-color:#f5f5f5;
}
```

17.6.2 缩放功能

zoom 属性可用来对内容放大或缩小。zoom 属性值可以是浮点数、百分数或者是关键字 normal，比如 150%表示为原来的 1.5 倍，关键字 normal 则表示元素正常大小。通常 zoom 属性可结合脚本语言应用，产生一些有趣的效果，请看下面这个示例：



```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>zoom 属性</title>
<script language="javascript" type="text/javascript">
window.attachEvent("onload", function(){
    var p1 = document.getElementById("p1");
    p1.attachEvent("onmouseover", function(event){
        event.srcElement.style.zoom = "200%";
    });
    p1.attachEvent("onmouseout", function(event){
        event.srcElement.style.zoom = "normal";
    });
});
</script>
<style type="text/css">
p{
    padding:10px;
    border:1px solid gray;
}
</style>
</head>

<body>
<p id="p1">请移动鼠标到这里。</p>
</body>
</html>
```

在以上代码中，JavaScript 脚本的作用是为 p 元素增加两个事件处理函数，当鼠标指针移到 p 元素之上时，onmouseover 事件触发，这时把 p 元素的 zoom 属性设置为 200%，而当鼠标指针离开 p 元素时，onmouseout 事件触发，zoom 属性被设为 normal，即原始大小。这样就实现了一种放大文字的效果(如图 17-37 所示)。

请移动鼠标到这里。

请移动鼠标到这里。

图 17-37　zoom 属性实现放大效果

延伸： 读者可以访问微软的 msdn 网站获取更多有关微软 CSS 扩展的信息(扩展属性会注有“This property is a Microsoft extension to CSS”)，网址如下：
http://msdn.microsoft.com/en-us/library/ms531205(vs.85).aspx

17.7 Mozilla 扩展

与微软对 CSS 扩展类似，Mozilla 扩展(Mozilla Extension)也是对 CSS 属性的一种扩展，它只适用于 Firefox 浏览器(本小节示例均使用 Firefox 浏览器进行展示)，扩展中的所有名称均以“-moz”开头。扩展提供了 at 规则、属性、属性值、伪元素和伪类等。除了可以完成一些 CSS 标准属性的功能外，还可以为 XUL(XML User-interface Language)等其他类型语言提供样式支持。

17.7.1 at 规则

前面我们接触过一些 at 规则，比如@media、@import。

Mozilla 扩展中也提供了一个 at 规则：@-moz-document，这个规则可以将样式限定于特定 URL 下的页面。比如：

```
@-moz-document url(http://www.huistd.com/),
            url-prefix(http://bbs.huistd.com/),
            domain(huistd.com)
{
    body{color:purple; background:yellow;}
}
```

通过设定@-moz-document，上面的 CSS 规则只用于满足如下条件的页面：

- http://www.huistd.com/所指向的页面。
- 任何以“http://bbs.huistd.com/”开头的 URL 所指的页面。
- 任何以“huistd.com”为主机的 URL 或以“.huistd.com”结束的 URL 所指向的页面。

17.7.2 伪类和伪元素

Mozilla 扩展提供了比 CSS 标准中更为丰富的伪类和伪元素，这里挑选其中的一部分来举例说明。

1. :-moz-list-bullet 伪类

:-moz-list-bullet 伪类可用来编辑 li 元素的标记，例如：

```
ul.ext li:-moz-list-bullet{
    font-size:30px;
    color:yellow;
}

<ul>
   <li>列表项 1</li>
   <li>列表项 2</li>
```



```
    <li>列表项 3</li>
</ul>
<ul class="ext">
    <li>列表项 1</li>
    <li>列表项 2</li>
    <li>列表项 3</li>
</ul>
```

第二个列表项的标记大小为 30px，颜色为黄色。在讲列表时提到，列表项标记的样式和该列表项中文字的样式一致，使用这个伪类可以给文字和标记提供不同的样式。效果如图 17-38 所示。

- 列表项1
- 列表项2
- 列表项3

- 列表项1
- 列表项2
- 列表项3

图 17-38　使用:-moz-list-bullet 伪类调整列表项标记样式

2. :-moz-first-node 和:-moz-last-node 伪类

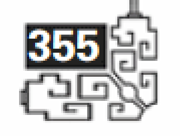
:-moz-first-node 和:-moz-last-node 伪类将分别匹配这样的元素：该元素是其父元素的第一个子结点(Node)以及该元素是其父元素的最后一个子结点。

请看示例：

```
span{
    display:block;
}
span:-moz-first-node{
    border:1px solid gray;
}
span:-moz-last-node{
    font-weight:bold;
    font-size:1.2em;
}

<div>
  <span>此 span 元素为 div 元素的第一个子结点。</span>
  <span>中间的 span 元素。</span>
  <span>中间的 span 元素。</span>
  <span>此 span 元素为 div 元素的最后一个子结点。</span>
</div>
```

作为 div 元素的第一个子结点与最后一个子结点的 span 元素被设定了不同的样式。效果如图 17-39 所示。



此span元素为div元素的第一个子结点。
中间的span元素。
中间的span元素。
此span元素为div元素的最后一个子结点。

图 17-39　:-moz-first-node 和:-moz-last-node 伪类的应用

17.7.3　属性

1. 更改元素背景区域范围

CSS 规范中规定元素背景区域的大小为边框外围的大小，也就是说边框会覆盖一部分背景区域。-moz-background-clip 属性可指定背景区域的范围，属性值 padding 表示背景区域以填充外侧为准，请看示例：

```
div{
    border:8px dashed black;
    background-color:yellow;
    width:100px;
    margin:10px;
    padding:10px;
    float:left;
}
div.clip {
    -moz-background-clip:padding;
}

<div>背景区域从边框位置开始</div>
<div class="clip">背景区域从填充位置开始</div>
```

效果如图 17-40 所示。

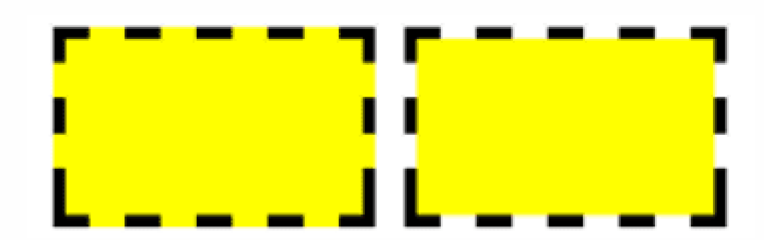


图 17-40　使用-moz-background-clip 属性更改背景区域范围

2. 圆角效果

-moz-border-radius 属性可用来给元素添加圆角效果，请看示例：

```
div.roundCorner{
    border:2px solid gray;
    width:200px;
    height:200px;
    -moz-border-radius:12px;
}

<div class="roundCorner"></div>
```



效果如图 17-41 所示，右图为将圆角放大后的效果，可以看出圆角含有锯齿。

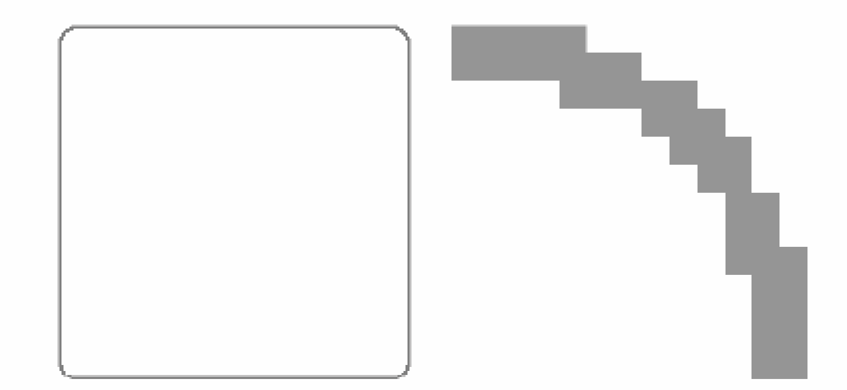


图 17-41　创造圆角效果

3．透明效果

与 IE 的 Alpha 滤镜类似，Mozilla 提供了-moz-opacity 属性用来控制元素的透明程度。此属性已经被 CSS3 规范支持，属性名称为 opacity，Firefox 同样支持这个属性。例如：

```
div#background{
    background:url(images/backwall.jpg);
    width:200px;
    height:120px;
}
div#inner{
    width:80px;
    padding:8px;
    background:yellow;
    opacity:0.4;
}

<div id="background"><div id="inner">此元素内容具有半透明效果。</div></div>
```

效果如图 17-42 所示，与 IE 的 Alpha 滤镜产生的效果是一致的。



图 17-42　使用 opacity 属性实现半透明效果

4．添加外轮廓线

-moz-outline 属性可以在元素的边框外侧再添加一层轮廓线。轮廓线的风格、颜色、宽度、与边框之间的距离(通过-moz-outline-offset 修改)以及是否需要圆角(通过-moz-outline-radius 修改)都可以控制。请看示例：

```
div.outline{
    width:100px;
    height:100px;
    padding:10px;
```

```
    margin:20px;
    border:6px solid blue;
    -moz-outline: 10px solid red;   /* 宽度 风格 颜色*/
    -moz-outline-radius:6px;        /* 圆角大小*/
    -moz-outline-offset:5px;        /* 偏移量*/
}

<div class="outline">div 元素</div>
```

以上代码将产生如图 17-43 所示的效果。

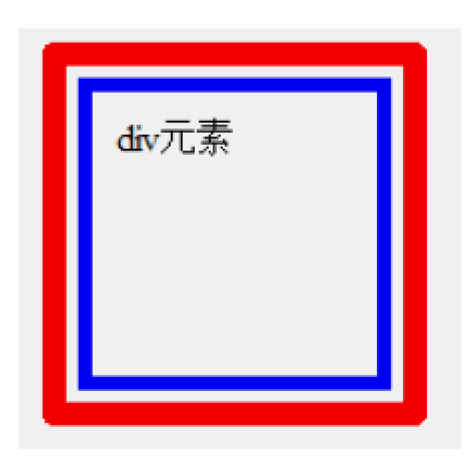


图 17-43　给元素增加外轮廓线

5. 焦点、选择与输入

下面将要介绍的属性会影响页面文本或控件的行为，-moz-user-input 属性可设定元素是否接受输入，-moz-user-select 属性可决定用户是否可以选择元素中的内容。比如下面的示例中，用户无法选择 p 元素内的文本，无法向文本框中输入任何内容：

```
p.noSelect{
    -moz-user-select:none;
}
input.noInput{
    -moz-user-input:disabled;
}

<p class="noSelect">这里的文本将无法选择</p>
<input class="noInput" type="text" value="这里将无法输入文本" />
```

17.7.4　属性值

1. display 属性值

display 属性规定了元素的显示方式，Mozilla 扩展中的包含了一些 display 属性值。前面说过 Firefox2 不支持 CSS 标准中的 inline-block 属性值，可以使用 Mozilla 扩展中的-moz-inline-block 来实现同等效果。

2. 表示颜色的关键字

颜色关键字的使用能够优化 CSS 代码的结构，如果希望页面某些元素的颜色和链接的文



字颜色一致，那么就可以使用属性值-moz-hyperlinktext，这样元素的颜色就和链接文字的颜色“绑定”在一起。如果链接文字的颜色变了，那么该元素的颜色也会随之更改，从而减少了修改工作。

3. 文本对齐

在 IE 中 text-align 属性不仅影响直接包含文本的元素的对齐方式，也影响其他类型的元素(比如 div、img 等)的对齐方式。但是 Firefox 中的 text-align 属性只能控制类似 p、span 一类直接包含文本的元素的对齐方式，而对于其他元素只能利用 margin 属性进行控制(参见盒模型部分的相关内容)。比如：

```
body{
    text-align:center;
}
div{
    width:300px;
    border:1px solid gray;
}

<p>在 Firefox 中，本行文本居中对齐</p>
<div><p>在 Firefox 中，本行文本不居中对齐。</p></div>
```

下面看看在 Firefox 浏览器中的显示效果(见图 17-44)。

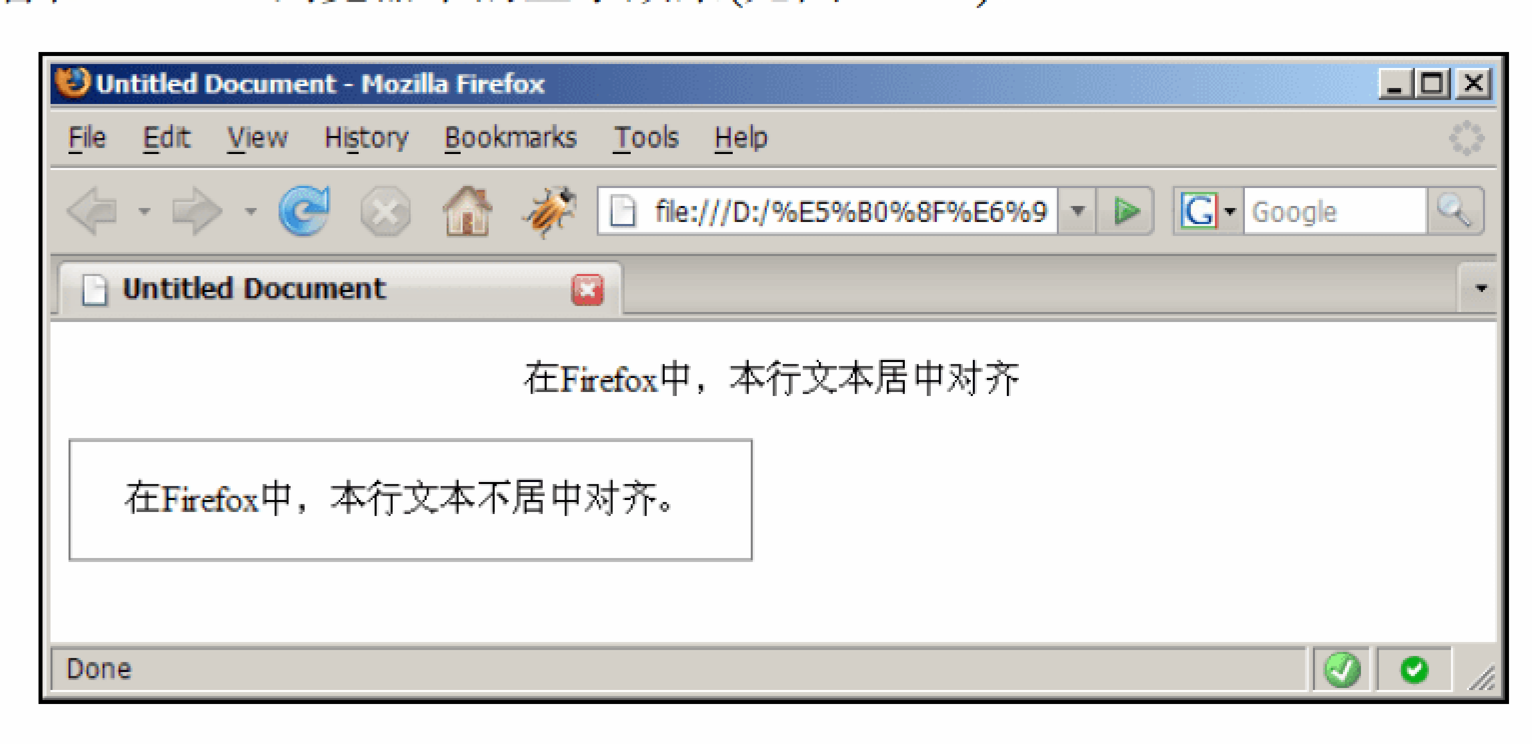


图 17-44　text-align 属性对 div 一类元素不起作用

Mozilla 扩展针对 text-align 属性提供了三个属性值：-moz-left、-moz-center 和-moz-right，使用这些属性值就可以控制其他任何元素的对齐方式。

假如我们将以上代码做如下修改：

```
body{
    text-align:-moz-center;
}
```

则 div 元素也会居中对齐，效果如图 17-45 所示。





图 17-45　使用扩展属性-moz-center 后，div 元素可实现居中对齐

4. 控制颜色

Mozilla 扩展中提供了两个添加颜色的函数，分别是-moz-hsla 和-moz-rgba，它们的用法如下：

- -moz-hsla(hue, saturation, lightness, alpha)
- -moz-rgba(red, green, blue, alpha)

第一个函数中的参数含义依次是色调、饱和度、亮度和透明度，前 3 个参数确定了颜色，最后一个确定了颜色的透明程度。多数读者可能对这种控制颜色的方式不了解，我们借助 Photoshop 软件中的调色板来进行讲解(见图 17-46)。

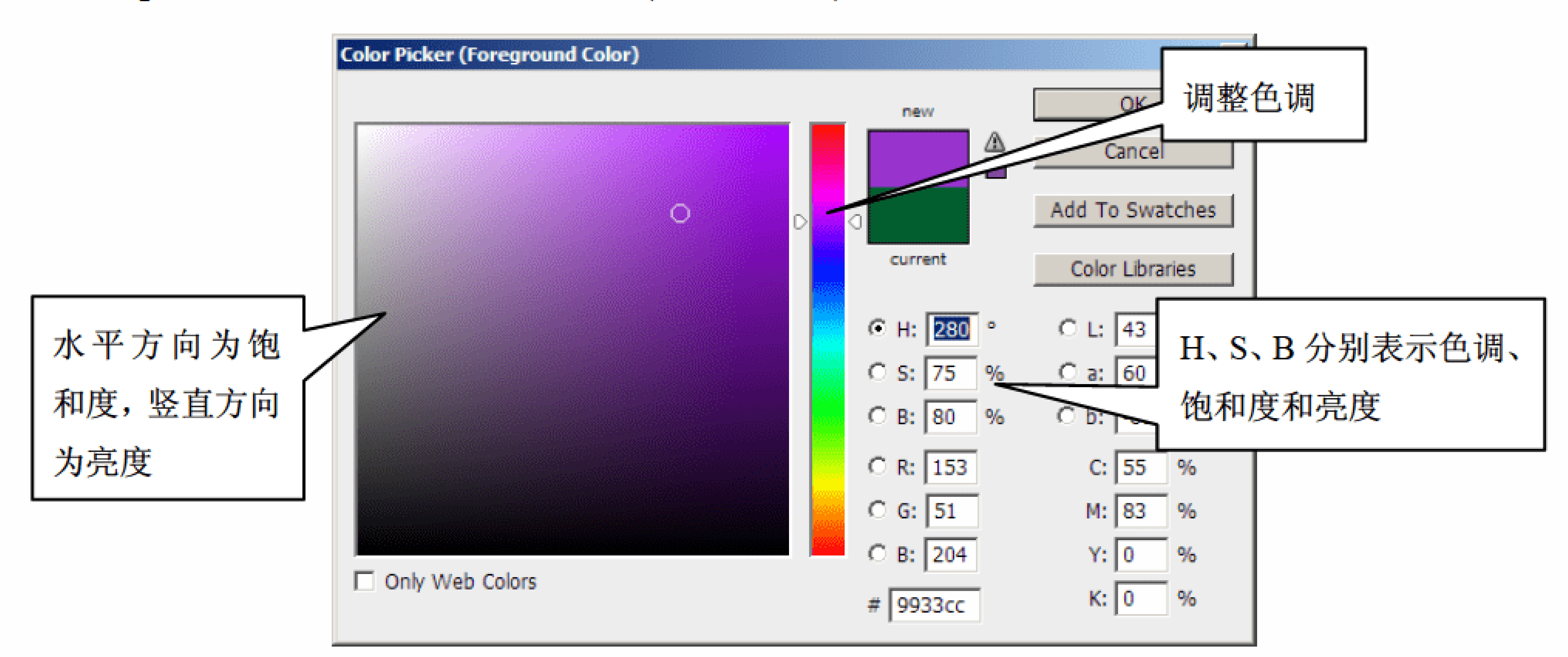


图 17-46　Photoshop 的调色板

色调(Hue)是区分不同色彩的名称。我们常说的红色、蓝色等就是最基本的色彩名称。色调取值范围为 0~359。饱和度(Saturation)是指颜色的纯正程度，饱和度越高色彩越纯，饱和度越低则颜色越接近灰色。饱和度取值范围为 0%~100%。亮度(Lightness)表示颜色的明亮程度，取值从 0%到 100%。

第 2 个函数中的参数依次是红色、绿色、蓝色和透明度。

目前 Firefox2 还不支持透明度的显示。

请看示例：



```
div{
    height:100px;
    width:180px;
}
div#div1{
    background:-moz-hsla(199, 85%, 93%, 100);
}
div#div2{
    background:-moz-rgba(234, 33, 99, 100);
}

<div id="div1"></div>
<div id="div2"></div>
```

效果如图 17-47 所示。

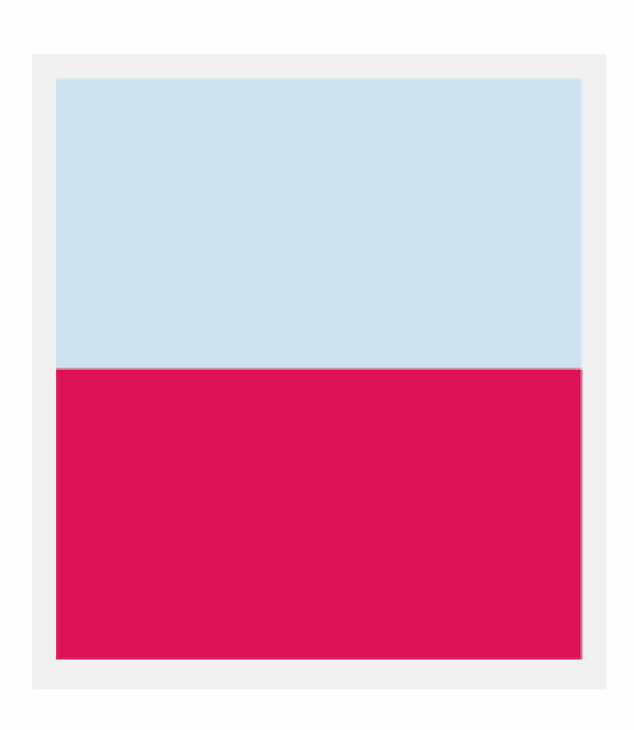


图 17-47　mozilla 扩展中的颜色函数

延伸： 目前，Mozilla 扩展中的部分属性已被提交至 W3C，作为 CSS3 规范中的标准属性。读者可以通过以下网址了解更多关于 Mozilla 扩展的信息：
http://developer.mozilla.org/en/docs/CSS_Reference:Mozilla_Extensions

17.8　小　　结

本章介绍了一些 CSS 的高级开发应用，内容包括 CSS 在 XML 文档中的应用、CSS 如何控制打印机的输出、CSS 和用户界面元素、滤镜、CSS 表达式、微软和 Mozilla 对 CSS 的扩展。

XML 是一种灵活、健壮的标记语言。一个 XML 文档包含序言和文档元素两部分，序言部分的 XML 文档声明指明文档的版本、字符集信息。XML 文档只允许有一个根元素，所有内容均包含在此根元素内。

使用 CSS 可以控制 XML 文档的显示。CSS 的 display 属性可以使 XML 元素以特定的(X)HTML 元素显示方式显示，比如块级元素、内联元素、表格、列表等。

CSS 的浮动和定位同样可以用于布局 XML 文档，与应用于(X)HTML 文档中没有两样。

尽管 CSS 可以用来给 XML 文档添加样式，但其效果与(X)HTML 存在一定的差距。各个

浏览器对 XML 文档的显示存在差异，且这种差异没有简单易行的办法来解决。因此大部分网站是不会使用 XML 加 CSS 进行构建的。使用 XSLT(可扩展样式表语言转换)可以将 XML 文档转换为 XHTML 文档，再使用 CSS 添加样式，可达到较好的效果。

CSS 能够单独控制 Web 页面在打印机上的输出样式，可通过 style 元素的 media 属性、CSS 的@media 规则和@import 规则的媒介限制来实现。

由打印机输出的多个页面是分离的，而显示器不存在这样的问题。针对具有分页效果的媒介，CSS 提供了控制分页的三个属性：page-break-before、page-break-after 和 page-break-inside，分别设置元素前、后和中间的分页方式。

鼠标指针、系统颜色、系统字体和轮廓线属于用户界面相关元素，CSS 提供了相应的样式属性，用来控制这些元素的样式。

鼠标指针可以提示用户正在或即将进行的操作的类型，cursor 属性可以更改元素的鼠标指针样式，除了使用 CSS 提供的关键字外，还可以使用自定义的鼠标指针。Firefox 浏览器可以使用 PNG、GIF 和 JPG 格式的图像，而 IE 只能使用后缀为 cur 的鼠标指针文件。Mozilla 扩展提供了额外的鼠标指针样式，它们可用于 Firefox 浏览器中。

如果设计者希望页面效果和用户的系统保持一致，那么 CSS 提供的系统颜色关键字可达到这个目的，每个系统颜色关键字对应着系统中某对象的颜色值。

轮廓线可以给元素添加一层线框，通常用在需要突出显示的元素上，CSS 的 outline 属性可以修改轮廓线样式，与边框不同的是，轮廓线不占用任何空间，且四个方向的轮廓线样式是一致的，不可单独进行调整。

IE 浏览器支持滤镜和转场，多个滤镜可以同时使用，使用滤镜可以解决 IE6 无法正常显示透明 PNG 图像的问题。转场一般结合 JavaScript 之类的脚本语言实现动态变化效果。

通过 behavior 属性，IE 可将 DHTML 的行为引入到页面中。IE 支持在 CSS 属性值中使用 JavaScript 代码，即 CSS 表达式。它可以动态地设定元素样式。但同时表达式也带来很严重的性能问题。

微软对 CSS 进行了扩展，其中包含了控制滚动条外观、缩放元素等功能的属性。

Mozilla 扩展是针对 Firefox 浏览器的，扩展提供了新的属性、属性值、颜色函数等。目前 Mozilla 扩展中的一部分属性已经被提交至 W3C，作为 CSS3 的标准属性。

下一章将进入本书的最后一部分内容——CSS 实战，将带领读者逐步完成一个比较完整的页面实例。



第六篇　CSS 实战

第 18 章　MyBlog 实例

本章将通过一个简单但较为完整的网站实例来介绍如何使用简洁的 XHTML 构建页面内容，并将所学的各种 CSS 样式运用到网站设计中。

18.1　实 例 说 明

本实例将带领读者完成一个常见的博客网站页面的制作，效果如图 18-1 所示。



图 18-1　MyBlog 的最终效果

这是一个假想的个人博客网站，叫做 MyBlog。页面以绿色调为主，白色调为辅。首页包含了标题、导航、侧栏、文章等内容。

18.2　从布局开始

18.2.1　结构分析

从图 18-1 中可以看出，在页面上部有标题和导航，中间部分的左侧显示了若干次要的栏目，右侧则显示部分文章内容，页面最低端为底脚区域，显示一些版权信息。很明显，可以用三行两列的布局方式来安排页面内容(见图 18-2)。

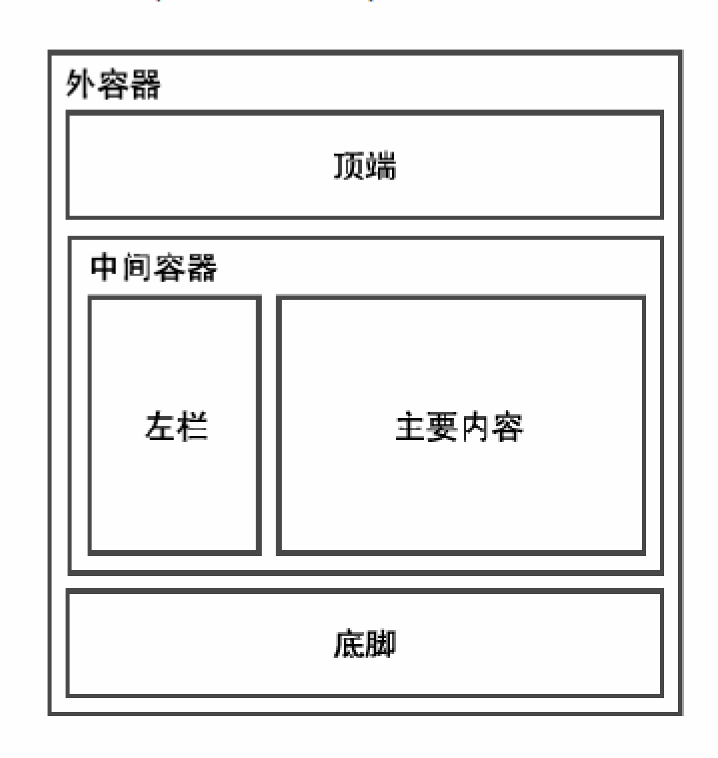


图 18-2　页面布局安排

竖直方向由上至下设置三个 div 元素，分别作为顶端、左栏和主要内容以及底脚的容器，然后在中间的 div 元素中设置两个 div 元素，分别作为左栏和右侧的主要内容。因为页面是水平居中对齐的，所以全部区域都放在一个外容器中，以便控制其对齐方式。

在分析完页面基本布局形式后，根据效果图，可初步确定各个区域所占空间的大小以及它们之间的距离，图 18-3 中标注了各区域的宽度值和它们的间距大小。

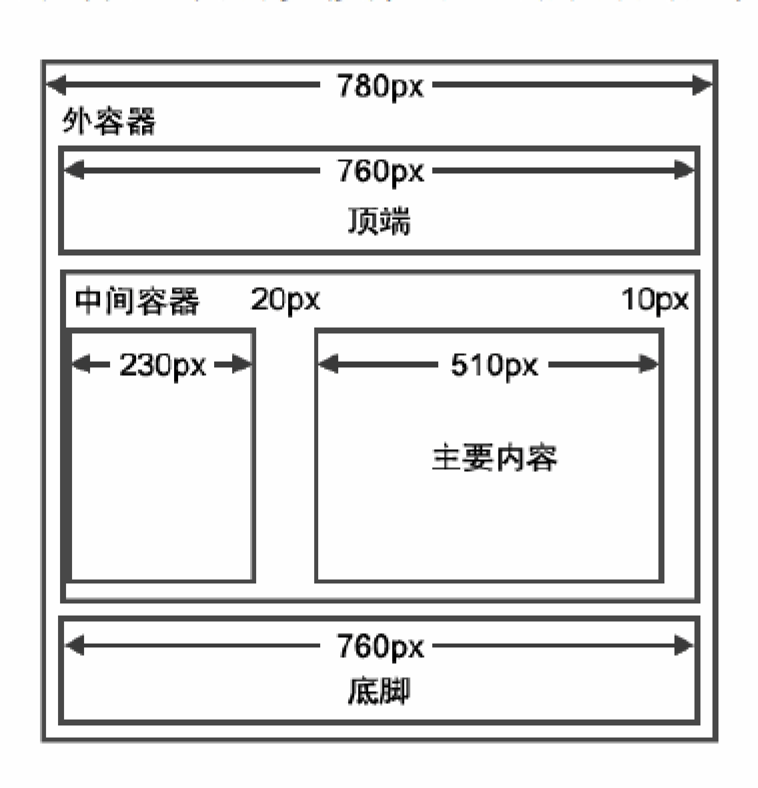


图 18-3　确定区域宽度及间距



18.2.2　准备 XHTML 代码

对布局进行分析后，就可以向文档中添加相应的 div 元素，作为每个区域的容器。

新建一个 html 文档，在 body 元素中输入以下代码，并保存为 index.html：

```
<div id="wrapper">
<div id="header"></div>
<div id="middle">
    <div id="sidebar"></div>
    <div id="content"></div>
</div>
<div id="footer"></div>
</div>
```

各个区域的 id 命名如代码所示。

18.3　准备样式表

将不同用途的 CSS 代码放在不同的样式表文件中有助于代码的管理和维护，对于本实例，我们将准备如下几个 CSS 文件。

- global.css：全局样式，用于整个站点的基本样式。
- index.css：首页样式。
- links.css：链接样式，应用于整个站点中出现的链接。
- iehack.css：IE 专用样式，用于处理浏览器显示差异以及兼容性问题。

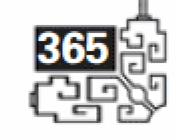
将以上 CSS 样式文件单独存放于一个目录下，取名为 css。另外，将页面所需的图片放置在 images 目录下。

在 index.html 中的 head 元素内添加 4 个 link 元素，将样式表文件引入到页面中。注意针对 IE 的样式 iehack.css 要写在条件注释中：

```
<link rel="stylesheet" type="text/css" href="css/global.css" />
<link rel="stylesheet" type="text/css" href="css/index.css" />
<link rel="stylesheet" type="text/css" href="css/link.css" />
<!--[if IE]>
<link rel="stylesheet" type="text/css" href="css/iehack.css" />
<![endif]-->
```

在全局样式表中，一部分代码用来取消元素在浏览器中的默认样式，另一部分用来给整个网站添加公共的样式。代码如下：

```
*{
    margin:0;
    padding:0;
}
```



```
body{
    font-size:14px;
    font-family:Times New Roman, "宋体", sans-serif;
    background:#909090;
}
ul li{
    list-style:none;
}
a img{
    border:none;
}
```

代码首先利用通用选择符将所有元素的边距和填充设为 0，然后定义 body 元素的字体大小、字体族和背景色。最后取消 li 元素默认的列表样式以及链接中图像所带有的边框。

页面尺寸大小已经确定，因此在 index.css 中添加如下代码：

```
div#wrapper{
    width:780px;
    margin:0 auto;
    background:#FFF;
}
```

#wrapper 宽度为 780px，背景色为白色，margin 属性的作用是使元素水平居中对齐。

在 link.css 中添加 a 元素的全局样式：

```
a{
    color:#26670D;
    text-decoration:none;
}
a:hover{
    text-decoration:underline;
}
```

延伸： 著名的 CSS 专家 Eric Meyer 编写了一个样式文件，重置了浏览器为元素增加的所有默认样式，读者可以访问 http://meyerweb.com/eric/tools/css/reset/，查看并下载这个文件。另外，Yahoo! User Interface Library (YUI)也提供了类似作用的样式表，读者可以访问 http://developer.yahoo.com/yui/以获得更多详细的信息。

18.4　添加标题和导航

18.4.1　准备 XHTML 代码

页面的顶部包含标题和导航两部分，为此我们在#header 中添加一个 h1 元素和一个 ul 元素，ul 中包含若干 li 元素，在 li 中放置链接：



```
<div id="header">
    <h1>My Blog</h1>
    <ul id="navi">
        <li id="home"><a href="">博客首页</a></li>
        <li id="article"><a href="">我的日志</a></li>
        <li id="photo"><a href="">我的相册</a></li>
        <li id="message"><a href="">给我留言</a></li>
    </ul>
</div>
```

可以注意到，标题和导航部分内容直接使用 XHTML 中相应的元素，我们没有再为它们额外添加容器器(比如 div)。除非页面效果需要这样的元素存在，否则这样做是画蛇添足，既破坏了代码的简洁性，又会影响页面的加载速度。

18.4.2　标题样式

首先准备一张图片，用作顶部区域的背景(见图 18-4)。

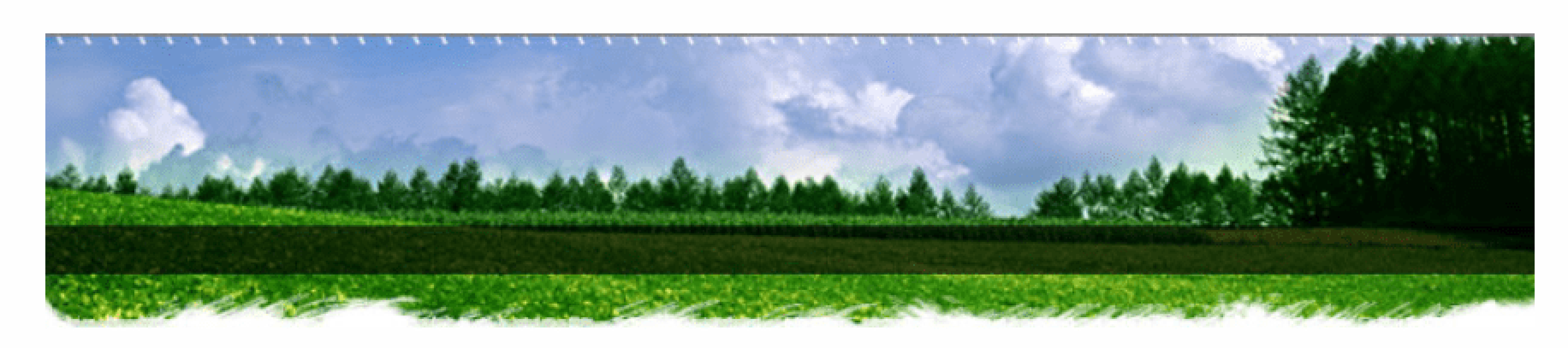


图 18-4　顶部区域的背景图

然后对#header 添加如下样式：

```
div#header{
    width:760px;
    background:url(../images/header_bg.jpg);
    height:150px;
    margin:0 auto;
}
```

页面顶端区域的高度是根据图片而定的，因此可设为固定值 150px。由于图片和样式文件单独放在各自的目录中，因此要注意 background 属性引用图像文件的路径是否正确。预览效果如图 18-5 所示。

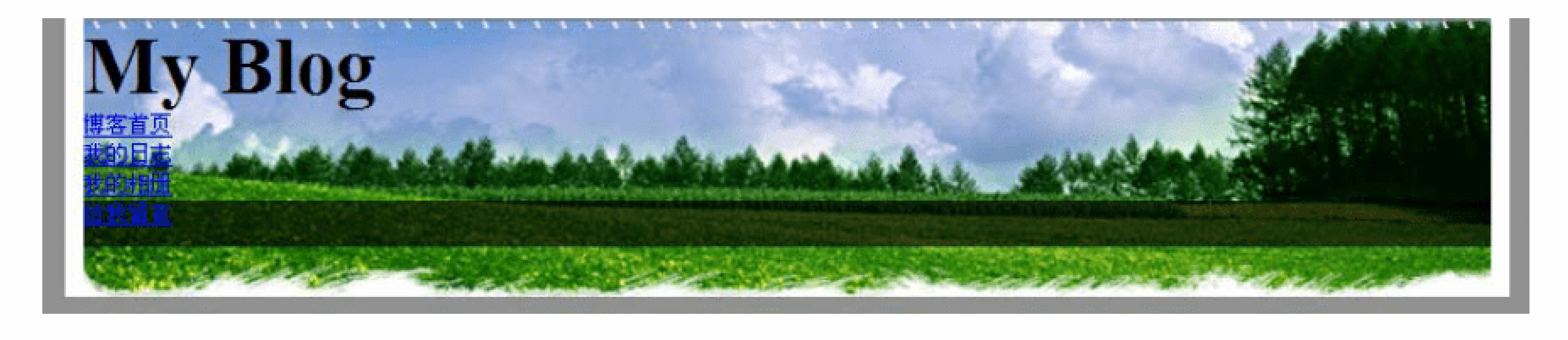
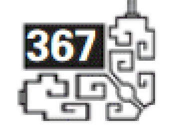


图 18-5　预览效果



接着对区域中的标题添加样式：

```
div#header h1{
    color:#000;
    background:#FFF;
    padding:3px 5px;
    margin:30px 0 0 40px;
    font-size:20px;
    float:left;
    border:1px solid #777;
    width:300px;
    display:inline;
}
```

h1 元素没有 id 属性，可通过后代选择符对其添加样式。以上代码设置了 h1 元素的文字颜色和大小、背景色以及盒模型相关的属性。从图 18-1 的效果看，标题有一定的透明度，于是再添加三条声明：

```
opacity:0.7;        /* for Firefox that support CSS3 */
-moz-opacity:0.7;   /* for Firefox */
filter:progid:DXImageTransform.Microsoft.Alpha(opacity="70"); /* for IE */
```

opacity 属性属于 CSS3 的标准，目前新版的 Firefox 浏览器已经支持这个属性，对于不支持该属性的 Firefox 可使用 Mozilla 扩展提供的-moz-opacity 属性，对于 IE 浏览器可使用滤镜。图 18-6 展示了当前效果。



图 18-6　半透明效果

18.4.3　导航设计

导航使用列表元素实现，首先定义 ul 元素的样式，CSS 代码如下：

```
div#header ul{
    float:right;
    width:320px;
    margin:100px 0 0 0;
    color:#FFF;
}
```

导航菜单是靠右对齐的，因此将它向右浮动。接着定义所有 li 元素的样式，代码如下：

```
div#header ul li{
    float:left;
    padding:3px 5px 3px 16px;
    margin-left:6px;
    background-position:left 5px;
    background-repeat:no-repeat;
}
```

float:left 声明将 li 元素排列成一行，background-position 和 background-repeat 属性定义了背景图像的位置和重复方式，下面通过 id 选择符添加导航菜单中每项之前的图标，代码如下：

```
li#home{
    background:url(../images/navi_home.gif);
}
li#article{
    background:url(../images/navi_article.gif);
}
li#photo{
    background:url(../images/navi_photo.gif);
}
li#message{
    background:url(../images/navi_message.gif);
}
```

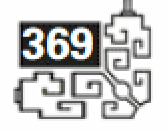
最后在 link.css 中添加 a 元素的样式，代码如下：

```
#header li a{
    color:#FFF;
    text-decoration:none;
}
#header li a:hover{
    color:#F6F797;
}
```

现在，在不同的浏览器中观察一下导航效果(见图 18-7)。



图 18-7　IE(上)和 Firefox(下)的显示效果



仔细观察会发现，导航在黑色区域中的位置不一致，IE 要比 Firefox 中的位置稍稍偏上一些，产生此问题的原因在于，IE 和 Firefox 对 padding 属性的处理方式存在差异，导致相同的填充和边距值产生不一致的效果。对此，我们在 iehack.css 中修改 li 元素的 padding 属性，使 IE 和 Firefox 应用不同的 padding 值，代码如下：

```
div#header ul li{
    padding-top:5px;
}
```

如图 18-8 所示为 IE 浏览器中的效果。



图 18-8　标题和导航的最终效果

18.5　左栏内容设计

18.5.1　处理#middle

#middle 是中间区域的容器，里面包含左栏和右侧的文章显示，在处理左栏之前先要为它添加样式，代码如下：

```
div#middle{
    margin:10px 0 0 0;
    padding:0 10px;
}
```

#middle 距离#header 有 10px 间距，且左右各有 10px 的填充。

18.5.2　处理左栏

左栏位于页面的中间靠左部分，里面包含三个栏目，首先添加三个栏目所需的 XHTML 代码，在#sidebar 中添加如下代码：

```
<div id="pinfo" class="sideBox">
    <h3>个人信息</h3>
    <a href="#"><img src="images/head.gif" alt="头像" /></a>
    <div class="item">注册时间: 2007-01-11</div>
    <div class="item">最近更新: 2008-03-12</div>
    <div class="item">最近登录: 2008-03-13</div>
</div>
```

```
<div id="recentMsg" class="sideBox">
    <h3>最新留言</h3>
    <div class="item">
        <span>网友</span><br />
        <a href="">随便看看~</a>
    </div>
    <div class="item">
        <span>Lynda</span><br />
        <a href="">看到你去巴黎的照片了。</a>
    </div>
    ...
    <div id="viewMsg" class="item">
        <a href="">查看全部留言 &gt;&gt;</a>
    </div>
</div>
<div id="stat" class="sideBox">
    <h3>访问统计</h3>
    <div class="item">近日访问: 1285 人</div>
    <div class="item">总访问量: 29834 人</div>
</div>
```

每个栏目都有个 div 元素作为容器，通过 id 对不同栏目进行标识，它们的 class 属性均为 sideBox。

18.5.3　添加样式

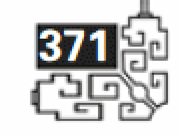
首先对#sidebar 添加样式，代码如下：

```
div#sidebar{
    float:left;
    width:230px;
    display:inline;     /* fix IE double-margin bug */
}
```

通过 class 选择符给栏目容器添加样式，代码如下：

```
div.sideBox{
    margin:0 0 10px 0;
    padding:0 0 8px 0;
    background:url(../images/line.gif) left bottom repeat-x;
}
```

每个栏目的容器底端被放置了背景图(line.gif，见图 18-9)，作为栏目之间的分隔线。



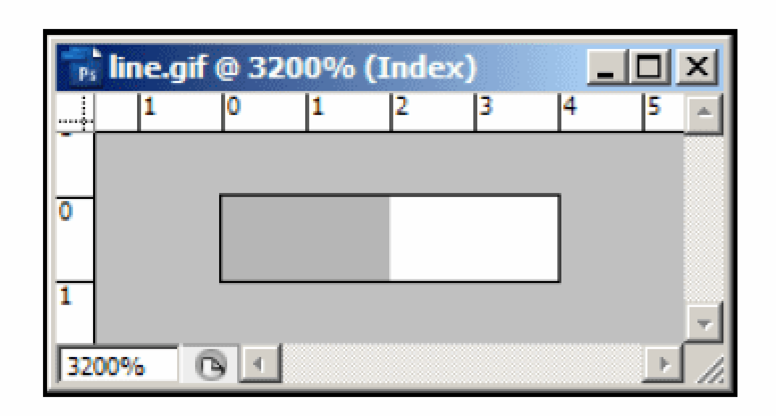


图 18-9　line.gif 作为栏目之间的分隔线

下面添加栏目标题的样式，代码如下：

```
div.sideBox h3{
    color:#FFF;
    font-size:12px;
    padding:8px 0 4px 20px;
    margin-bottom:10px;
    width:210px;
    background:url(../images/sidebox_header.jpg) left top no-repeat;
}
```

由于 IE 和 Firefox 对 padding 解析的不同，将以下代码添加到 iehack.css 中：

```
div.sideBox h3{
    padding-top:10px;
}
```

图 18-10 展示了当前页面的效果。



图 18-10　侧栏的初步效果

添加如下代码来控制个人信息中的头像图片：



```
div#pinfo a img{
    margin:10px 0 5px 20px;
    padding:3px;
    border:1px solid #909090;
}
```

每个栏目中的信息都被包含在 class 属性为 item 的 div 元素中，给该元素添加如下样式：

```
div.item{
     margin:5px 10px 0 20px;
}
```

最新留言栏目中的条目要进行一些特殊处理，修改行间距、增加分隔线，代码如下：

```
div#recentMsg div.item{
    line-height:1.5em;
    margin-left:15px;
    padding-left:5px;
    border-bottom:1px solid #78AD27;
}
div#recentMsg div#viewMsg{
    border:none;
}
```

现在预览一下左栏的效果(见图 18-11)。



图 18-11　左栏的最终效果



18.6 文章显示

18.6.1 处理#content

#content 是页面中间右侧区域的容器，首先给#content 添加样式，设置整个区域：

```
div#content{
    float:left;
    margin-left:20px;
    width:510px;
}
```

#content 向左浮动，与#sidebar 之间有 20px 的距离，宽度为 510px，与如图 18-3 所示的宽度大小相一致。

由于#middle 中的两个子元素#sidebar 和#content 全部采用浮动布局，因此它的高度将不会随着这两个元素进行变化，所以我们要在#content 之后添加一个 div 元素，用来清除浮动：

```
<div id="middle">
    <div id="sidebar">
    ...
    </div>
    <div id="content"></div>
    <div class="clearFloat"></div>
</div>
```

相应地，在 global.css 中添加如下代码：

```
.clearFloat{
    clear:both;
}
```

18.6.2 准备代码

在#content 中添加如下 XHTML 代码：

```
<div class="contentBox">
    <h2>到家了</h2>
    <p>昨晚刚从成都回到北京，回家倒头便睡，一觉醒来已经临近中午了。随便吃了些东西就习惯性
地坐在了电脑前面。</p>
    <p>出去了这么久，很长时间没来更新了，先发些照片吧。
        <span class="imgBox">
            <img src="images/citynight.jpg" alt="城市夜景" />
        </span>
        <span class="imgBox">
            <img src="images/moutain.jpg" alt="山" />
```



```
        </span>
    </p>
    <a href="" class="floatRight">阅读全文 &gt;&gt;</a>
    <div class="artInfo">2008-03-12 | 评论(3) | 阅读(215)</div>
    <div class="clearFloat"></div>
</div>
<div class="contentBox">
    <h2>无题</h2>
    <p>感觉最近总想写些什么，脑子里充满了各种奇特的想法。但是一坐到电脑前面，双手放在键盘
上时，那些奇思妙想突然就消失的无影无踪了，结果竟然一个字都没有写。什么时候能发明一种机器，
能直接将想法转成文字，不用费劲敲键盘该多好。</p>
    <a href="" class="floatRight">阅读全文 &gt;&gt;</a>
    <div class="artInfo">2007-12-17 | 评论(12) | 阅读(1376)</div>
    <div class="clearFloat"></div>
</div>
```

class 属性为 contentBox 的 div 元素作为每篇文章的容器，其中 h2 包含文章标题，p 元素包含文章内容，最后的 a 元素和 div 元素用来放置其他信息。

18.6.3　添加样式

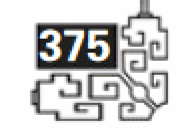
首先对文章的容器添加样式，代码如下：

```
div.contentBox{
    margin:0px 10px 20px 0;
    background:url(../images/line.gif) left bottom repeat-x;
}
```

文章之间用虚线进行分隔，在区域底部添加水平方向重复出现的分隔线。

接着给文章标题、文章段落以及段落中的图片容器添加样式，代码如下：

```
div.contentBox h2{
    color:#2E6722;
    font-size:14px;
    padding:6px 0 0 10px;
    margin:0 0 20px 0;
}
div.contentBox p{
    font-size:14px;
    text-indent:2.2em;
    letter-spacing:0.1em;
    line-height:1.3em;
    color:#333333;
    margin:20px 0;
}
div.contentBox span.imgBox{
    display:block;
    text-indent:0;
    margin:20px 0;
    text-align:center;
}
```



段落中文字的行间距为 1.3em，字符之间的距离为 0.1em，这样可防止文字过于密集，另外适当增加字符间距也能提高文字的可阅读性。由于每个字符之间多增加了 0.1em 的距离，因此段落首行缩进为 2.2em。.imgBox 中声明“display:block;”使 span 元素以块级元素方式显示，声明“text-indent:0;”取消其继承的缩进值，声明“text-align:center;”使图片水平居中对齐。

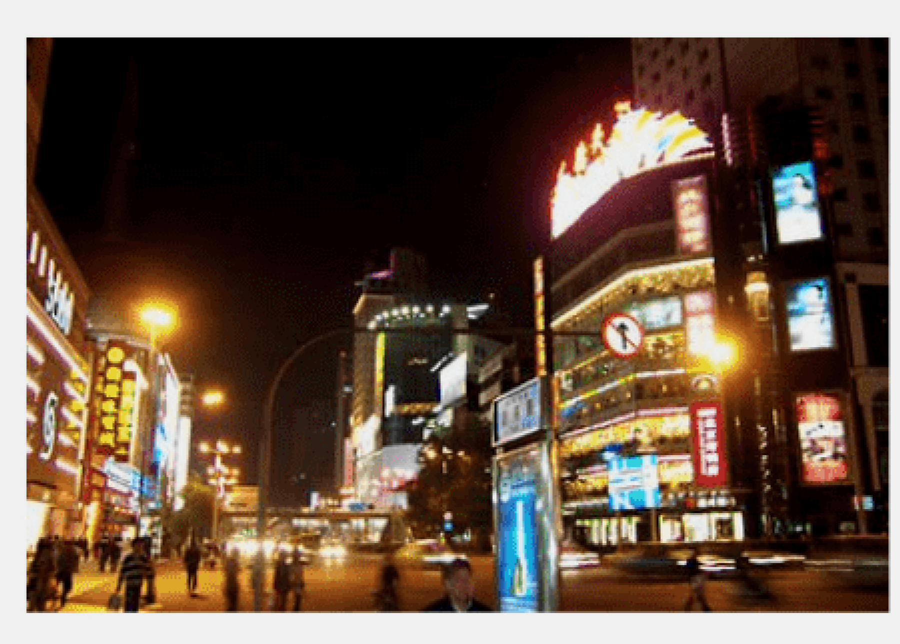
提示： 由于 p 元素中不能再包含其他块级元素，所以这里使用 span 元素作为 img 的容器。另外，如果不给 img 元素添加容器，使用如下方法也能实现图像的水平居中：给 img 元素添加声明“margin:20px auto;”，利用边距的 auto 值来实现水平居中，但前提是 img 的父元素需要有确定的宽度值。

如图 18-12 所示为页面的效果。

到家了

昨晚刚从成都回到北京，回家倒头便睡，一觉醒来已经临近中午了。随便吃了些东西就习惯性地坐在了电脑前面。

出去了这么久，很长时间没来更新了，先发些照片吧。





阅读全文 >>
2008-03-12 | 评论(3) | 阅读(215)

无题

感觉最近总想写些什么，脑子里充满了各种奇特的想法。但是一坐到电脑前面，双手放在键盘上时，那些奇思妙想突然就消失的无影无踪了，结果竟然一个字都没有写。什么时候能发明一种机器，能直接将想法转成文字，不用费劲敲键盘该多好。

阅读全文 >>
2007-12-17 | 评论(12) | 阅读(1376)

图 18-12　文章显示的效果



最后给文章的附加信息添加样式，使其向右对齐：

```
div.artInfo{
    float:right;
    width:200px;
}
```

元素的浮动导致它将脱离父元素的包围，这时效果会如图 18-13 所示。



图 18-13　文章时间等信息位于分隔线之下

在该元素后添加清除浮动的 div 元素，使该元素能位于分隔线之内：

```
<div class="contentBox">
    <h2>到家了</h2>
    ...
    <div class="artInfo">2008-03-12 | 评论(3) | 阅读(215)</div>
    <div class="clearFloat"></div>
</div>
<div class="contentBox">
    ...
    <div class="artInfo">2007-12-17 | 评论(12) | 阅读(1376)</div>
    <div class="clearFloat"></div>
</div>
```

如图 18-14 所示为 Firefox 中显示的效果，文章日期等信息已经位于分隔线之内了。



图 18-14　修改后的页面效果(Firefox 浏览器)

如果读者使用 IE 浏览器观察效果，会发现一个奇怪的问题，下一篇文章的标题也跑到分隔线之上了(见图 18-15)。



图 18-15　IE 浏览器的问题(下一篇文章的标题位于分隔线之上)

这又是 IE 浏览器中的浮动问题，它是由浮动属性和背景属性共同产生的。在 iehack.css 添加如下样式规则可解决这个问题：

```
div.contentBox{
    height:1%;
}
```

最后在给.artInfo 增加一些下边距：

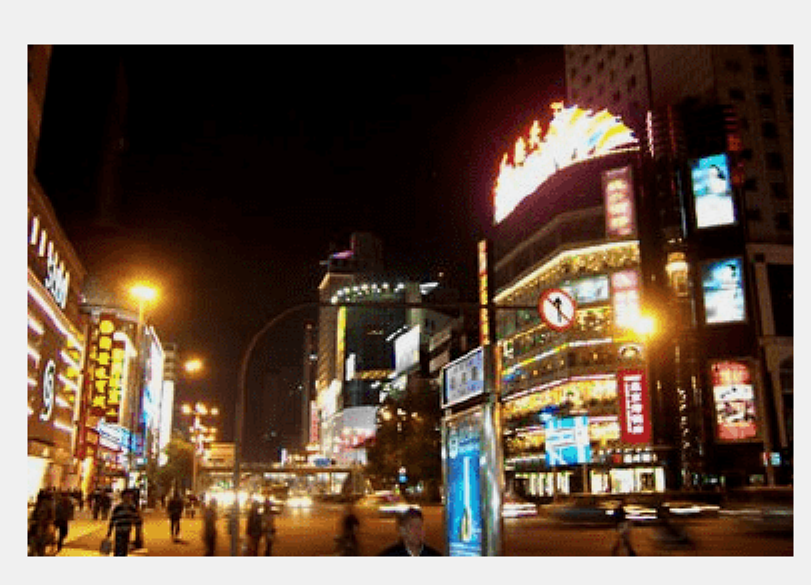
```
div.artInfo{
    float:right;
    width:200px;
    margin-bottom:10px;
}
```

现在页面效果如图 18-16 所示。

到家了

昨晚刚从成都回到北京，回家倒头便睡，一觉醒来已经临近中午了。随便吃了些东西就习惯性地坐在了电脑前面。

出去了这么久，很长时间没来更新了，先发些照片吧。





阅读全文 >>

2008-03-12 | 评论(3) | 阅读(215)

无题

感觉最近总想写些什么，脑子里充满了各种奇特的想法。但是一坐到电脑前面，双手放在键盘上时，那些奇思妙想突然就消失的无影无踪了，结果竟然一个字都没有写。什么时候能发明一种机器，能直接将想法转成文字，不用费劲敲键盘该多好。

阅读全文 >>

2007-12-17 | 评论(12) | 阅读(1376)

图 18-16　文章显示区域的最终效果

18.7　留 言 表 单

18.7.1　XHTML 代码

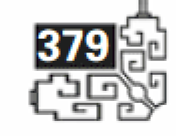
文章显示区域的下方提供了一个留言表单，表单的 XHTML 代码如下：

```
<div id="leaveMsg" class="contentBox">
    <form action="#" method="post">
    <h2>留言</h2>
    <table id="newMsg">
        <col id="tblHeader" />
        <col id="tblContent" />
        <tr>
            <td>姓名</td>
            <td><input type="text" tabindex="1" id="username" /></td>
        </tr>
        <tr>
            <td>电子信箱</td>
            <td><input type="text" tabindex="2" id="email" />(选填)</td>
        </tr>
        <tr>
            <td>内容</td>
            <td>
                <textarea cols="50" tabindex="3" rows="5" id="msgContent"></textarea>
            </td>
        </tr>
        <tr>
            <td></td>
            <td><input type="submit" tabindex="4" value="留  言" /></td>
        </tr>
    </table>
    </form>
</div>
```

18.7.2　添加样式

留言的标题应该和上面的文章标题有所区别，于是添加如下样式：

```
div#leaveMsg h2{
    font-size:12px;
    color:#454545;
    padding:0;
    margin:10px 0;
}
```



id 选择符比前面的属性选择符(注意 class 选择符属于属性选择符)的确定度高，因此留言区域的标题将应用这个样式。

对布局表单元素的 table 添加如下样式：

```
table#newMsg{
    margin-bottom:20px;
}
table#newMsg td{
    padding:2px 0;
}
col#tblHeader{
    width:80px;
}
```

表单元素的样式如下：

```
input, textarea{
    font-size:12px;
    font-family:Tahoma, "宋体", serif;
    border:1px solid #909090;
    color:#454545;
}
```

以上代码可添加到 global.css 中，作为所有表单元素的全局样式。其余样式如下：

```
input#username, input#email{
    width:160px;
    height:16px;
    padding:2px;
    background:#EFEFEF;
}
textarea#msgContent{
    width:300px;
    padding:2px;
    background:#EFEFEF;
}
```

图 18-17 展示了表单的最终效果。



图 18-17 留言表单的效果



18.8　底 脚 处 理

页面底脚含有一些版权声明的文字，其 XHTML 代码如下：

```
<div id="footer">
    <p>Copyright &copy; 2006 - 2008 Hui Studio, All Rights Reserved.</p>
    <p>小晖子工作室 版权所有</p>
</div>
```

样式代码如下：

```
div#footer{
    margin:30px 10px 0 10px;
    padding:10px 0;
    border-top:1px solid #CDCDCD;
}
div#footer p{
    text-align:center;
    margin:8px 0;
}
```

效果如图 18-18 所示。

Copyright © 2006 - 2008 Hui Studio, All Rights Reserved.
小晖子工作室 版权所有

图 18-18　底脚最终效果

至此，页面所有内容均已制作完成。页面的最终效果请参考图 18-1(Firefox 浏览器)。

18.9　完 整 代 码

本小节将提供一份本实例的完整代码。

18.9.1　XHTML 文档代码

index.html 的完整代码如下：

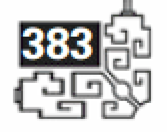
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>MyBlog</title>
```

```
<link rel="stylesheet" type="text/css" href="css/global.css" />
<link rel="stylesheet" type="text/css" href="css/index.css" />
<link rel="stylesheet" type="text/css" href="css/link.css" />
<!--[if IE]>
<link rel="stylesheet" type="text/css" href="css/iehack.css" />
<![endif]-->
</head>

<body>
<div id="wrapper">
<!-- header -->
<div id="header">
    <h1>My Blog</h1>
    <ul id="navi">
        <li id="home"><a href="">博客首页</a></li>
        <li id="article"><a href="">我的日志</a></li>
        <li id="photo"><a href="">我的相册</a></li>
        <li id="message"><a href="">给我留言</a></li>
    </ul>
</div>
<!-- middle -->
<div id="middle">
    <div id="sidebar">
        <div id="pinfo" class="sideBox">
            <h3>个人信息</h3>
            <a href="#"><img src="images/head.gif" alt="头像" /></a>
            <div class="item">注册时间: 2007-01-11</div>
            <div class="item">最近更新: 2008-03-12</div>
            <div class="item">最近登录: 2008-03-13</div>
        </div>
        <div id="recentMsg" class="sideBox">
            <h3>最新留言</h3>
            <div class="item">
                <span>网友</span><br />
                <a href="">随便看看~</a>
            </div>
            <div class="item">
                <span>Lynda</span><br />
                <a href="">看到你去巴黎的照片了。</a>
            </div>
            <div class="item">
                <span>Amy</span><br />
                <a href="">好久没有更新了吧。</a>
            </div>
            <div class="item">
                <span>网友</span><br />
                <a href="">不错</a>
            </div>
            <div class="item">
```



```
                <span>小丸子</span><br />
                <a href="">生日快乐啊。</a>
            </div>
            <div id="viewMsg" class="item">
                <a href="">查看全部留言 &gt;&gt;</a>
            </div>
        </div>
        <div id="stat" class="sideBox">
            <h3>访问统计</h3>
            <div class="item">近日访问: 1285 人</div>
            <div class="item">总访问量: 29834 人</div>
        </div>
    </div>
    <div id="content">
        <div class="contentBox">
            <h2>到家了</h2>
            <p>昨晚刚从成都回到北京，回家倒头便睡，一觉醒来已经临近中午了。随便吃了些东
               西就习惯性地坐在了电脑前面。</p>
            <p>出去了这么久，很长时间没来更新了，先发些照片吧。
                <span class="imgBox">
                    <img src="images/citynight.jpg" alt="城市夜景" />
                </span>
                <span class="imgBox">
                    <img src="images/moutain.jpg" alt="山" />
                </span>
            </p>
            <a href="" class="floatRight">阅读全文 &gt;&gt;</a>
            <div class="artInfo">2008-03-12 | 评论(3) | 阅读(215)</div>
            <div class="clearFloat"></div>
        </div>
        <div class="contentBox">
            <h2>无题</h2>
            <p>感觉最近总想写些什么，脑子里充满了各种奇特的想法。但是一坐到电脑前面，双
               手放在键盘上时，那些奇思妙想突然就消失的无影无踪了，结果竟然一个字都没有
               写。什么时候能发明一种机器，能直接将想法转成文字，不用费劲敲键盘该多好。
            </p>
            <a href="" class="floatRight">阅读全文 &gt;&gt;</a>
            <div class="artInfo">2007-12-17 | 评论(12) | 阅读(1376)</div>
            <div class="clearFloat"></div>
        </div>
        <div id="leaveMsg" class="contentBox">
            <h2>留言</h2>
            <form action="#" method="post">
            <table id="newMsg">
                <col id="tblHeader" />
                <col id="tblContent" />
                <tr>
                    <td>姓名</td>
                    <td><input type="text" tabindex="1" id="username" /></td>
```



```
                </tr>
                <tr>
                    <td>电子信箱</td>
                    <td><input type="text" tabindex="2" id="email" />(选填)
                    </td>
                </tr>
                <tr>
                    <td>内容</td>
                    <td>
                    <textarea cols="50" rows="5" tabindex="3" id="msgContent">
                    </textarea>
                    </td>
                </tr>
                <tr>
                    <td></td>
                    <td><input type="submit" tabindex="4" value="留  言" />
                    </td>
                </tr>
            </table>
            </form>
        </div>
    </div>
    <div class="clearFloat"></div>
</div>
<!-- footer -->
<div id="footer">
    <p>Copyright &copy; 2006 - 2008 Hui Studio, All Rights Reserved.</p>
    <p>小晖子工作室 版权所有</p>
</div>
</div>
</body>
</html>
```

18.9.2 样式表文档代码

global.css：

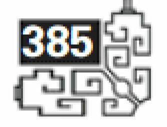
```
/**************************************************
 *  Created date:   2008-02-27
 *  Author:         Jia Zheng
 *  Project:        MyBlog
 *  Description:    eliminate default styles,
 *                  global styles and common
 *                  styles.
 **************************************************/
*{
    margin:0;
    padding:0;
```



```
}
body{
    font-size:12px;
    font-family:Times New Roman, "宋体", sans-serif;
    background:#909090;
}
ul li{
    list-style:none;
}
a img{
    border:none;
}
.clearFloat{
    clear:both;
}
input, textarea{
    font-size:12px;
    font-family:Tahoma, "宋体", serif;
    border:1px solid #909090;
    color:#454545;
}
```

index.css：

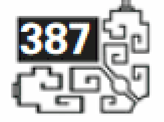
```
/*****************************************
 *  Created date:   2008-02-27
 *  Author:         Jia Zheng
 *  Project:        index page (MyBlog)
 *****************************************/
div#wrapper{
    width:780px;
    margin:0 auto;
    background:#FFF;
}
/****************
 *   header
 ****************/
div#header{
    width:760px;
    background:url(../images/header bg.jpg);
    height:150px;
    margin:0 auto;
}
div#header h1{
    color:#000;
    background:#FFF;
    padding:3px 5px;
    margin:30px 0 0 40px;
    font-size:20px;
    float:left;
```

```
    border:1px solid #777;
    width:300px;
    display:inline;
    opacity:0.7;        /* for Firefox that support CSS3 */
    -moz-opacity:0.7;   /* for Firefox */
    filter:progid:DXImageTransform.Microsoft.Alpha(opacity="70"); /* for IE
*/
}
div#header ul{
    float:right;
    width:320px;
    margin:100px 0 0 0;
    color:#FFF;
}
div#header ul li{
    float:left;
    padding:3px 5px 3px 16px;
    margin-left:6px;
    background-position:left 5px;
    background-repeat:no-repeat;
}
li#home{
    background:url(../images/navi home.gif);
}
li#article{
    background:url(../images/navi_article.gif);
}
li#photo{
    background:url(../images/navi photo.gif);
}
li#message{
    background:url(../images/navi message.gif);
}
/****************
 *   middle
 ****************/
div#middle{
    margin:10px 0 0 0;
    padding:0 10px;
}
/****************
 *   sidebar
 ****************/
div#sidebar{
    float:left;
    width:230px;
    display:inline;
}
div.sideBox{
```



```
    margin:0 0 10px 0;
    padding:0 0 8px 0;
    background:url(../images/line.gif) left bottom repeat-x;
}
div.sideBox h3{
    color:#FFF;
    font-size:12px;
    padding:8px 0 4px 20px;
    margin-bottom:10px;
    width:210px;
    background:url(../images/sidebox header.jpg) left top no-repeat;
}
div#pinfo a img{
    margin:10px 0 5px 20px;
    padding:3px;
    border:1px solid #909090;
}
div.item{
     margin:5px 10px 0 20px;
}
div#recentMsg div.item{
    line-height:1.5em;
    margin-left:15px;
    padding-left:5px;
    border-bottom:1px solid #78AD27;
}
div#recentMsg div#viewMsg{
    border:none;
}
/****************
 *   content
 ****************/
div#content{
    float:left;
    margin-left:20px;
    width:510px;
}
div.contentBox{
    margin:0px 10px 20px 0;
    background:url(../images/line.gif) left bottom repeat-x;
}
div.contentBox h2{
    color:#2E6722;
    font-size:14px;
    padding:6px 0 0 10px;
    margin:0 0 20px 0;
}
div.contentBox p{
    font-size:14px;
```



```
    text-indent:2.2em;
    letter-spacing:0.1em;
    line-height:1.3em;
    color:#333333;
    margin:20px 0;
}
div.contentBox span.imgBox{
    display:block;
    text-indent:0;
    margin:20px 0;
    text-align:center;
}
div.artInfo{
    float:right;
    width:200px;
    margin-bottom:10px;
}
/****************
 * message form
 ****************/
div#leaveMsg h2{
    font-size:12px;
    color:#454545;
    padding:0;
    margin:10px 0;
}
table#newMsg{
    margin-bottom:20px;
}
table#newMsg td{
    padding:2px 0;
}
col#tblHeader{
    width:80px;
}
input#username, input#email{
    width:160px;
    height:16px;
    padding:2px;
    background:#EFEFEF;
}
textarea#msgContent{
    width:300px;
    padding:2px;
    background:#EFEFEF;
}
/****************
 *   footer
 ****************/
```



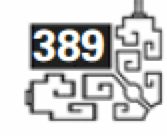
```
div#footer{
    margin:30px 10px 0 10px;
    padding:10px 0;
    border-top:1px solid #CDCDCD;
}
div#footer p{
    text-align:center;
    margin:8px 0;
}
```

link.css：

```
/*****************************************
 *  Created date:   2008-02-27
 *  Author:         Jia Zheng
 *  Project:        MyBlog
 *  Description:    styles for all links
 *****************************************/
a{
    color:#26670D;
    text-decoration:none;
}
a:hover{
    text-decoration:underline;
}
#header li a{
    color:#FFF;
    text-decoration:none;
}
#header li a:hover{
    color:#F6F797;
}
```

iehack.css：

```
/***********************************************
 *  Created date:   2008-02-27
 *  Author:         Jia Zheng
 *  Project:        MyBlog
 *  Description:    fix IE bugs
 ***********************************************/
 div#header ul li{
    padding-top:5px;
}
div.sideBox h3{
    padding-top:10px;
}
div.contentBox{
    height:1%;
}
```



18.10 小　　结

本章通过一个完整的页面实例，将本书所学的 CSS 知识融汇在一起。本实例除了涉及选择符、盒模型、控制表格、表单、列表元素外观、添加背景图像、页面布局等 CSS 基本内容外，还使用了 IE 中的滤镜和 Firefox 支持的 Mozilla 扩展，最后我们还解决了一个 IE 浮动所带来的显示问题。

当然，本实例所提供的方法并不是唯一的，读者可尝试使用其他方法重做本实例。

最后希望读者在实际开发中能够合理地运用 XHTML 和 CSS，用最简洁的代码实现最理想的效果。



参 考 文 献

1. CSS: The Definitive Guide, 3rd Edition, Eric Meyer, O'Reilly 2006
2. CSS Cookbook, Christopher Schmitt, O'Reilly 2004
3. CSS Mastery: Advanced Web Standards Solutions, Andy Budd, Cameron Moll, Simon Collison, Friendsof 2006
4. CSS: The Missing Manual, David Sawyer McFarland, O'Reilly 2006
5. Beginning CSS Webvelopment From Novice To Professional, Simon Collison, Apress 2006
6. Beginning CSS: Cascading Style Sheets for Web Design, 2nd Edition, Richard York, Wrox Press 2007
7. Beginning HTML with CSS and XHTML: Modern Guide and Reference, David Schultz, Craig Cook, Apress 2007
8. More Eric Meyer on CSS, Eric A. Meyer, New Riders Publishing 2004
9. Cascading Style Sheets: Designing for the Web, 3rd Edition, Hakon Wium Lie, Bert Bos, Addison Wesley Professional 2005
10. The Art & Science of CSS, Cameron Adams, Jina Bolton, David Johnson, Steve Smith, Jonathan Snook, SitePoint 2007
11. The Best-Practice Guide to XHTML&CSS, Patrick Griffiths, New Riders Publishing 2007
12. CSS Instant Results, Richard York, Wrox Press 2006
13. Pro CSS Techniques, Jeff Croft, Ian Lloyd, Dan Rubin, Apress 2006
14. Cascading Style Sheets Level 2 Revision 1(CSS 2.1) Specification, W3C Candidate Recommendation 19 July 2007
15. Cascading Style Sheets, level 1, W3C Recommendation 17 Dec 1996, revised 11 Jan 1999
16. XHTML 1.0 The Extensible HyperText Markup Language(Second Edition), A Reformulation of HTML 4 in XML 1.0, W3C Recommendation 26 January 2000, revised 1 August 2002
17. The Zen of CSS Design: Visual Enlightenment for the Web(Voices That Matter), Dave Shea, Molly E. Holzschlag, Peachpit Press 2005
18. Cascading Style Sheets 2.0 Programmer's Reference, Eric A. Meyer, McGraw-Hill 2001
19. CSS 入门经典(Teach Yourself CSS in 24 Hours, 2nd Edition, Kynn Bartlett, Sams Publishing 2006 中文版)，周哲，亲玉等译，人民邮电出版社 2007
20. CSS 网站布局实录：基于 Web 标准的网站设计指南，第 2 版，李超，科学出版社 2007
21. 网站重构：应用 Web 标准进行设计(Designing with Web Standards, Jeffrey Zeldman, New Riders Publishing 2004 中文版)，傅捷，王宗义，祝军译，电子工业出版社 2004